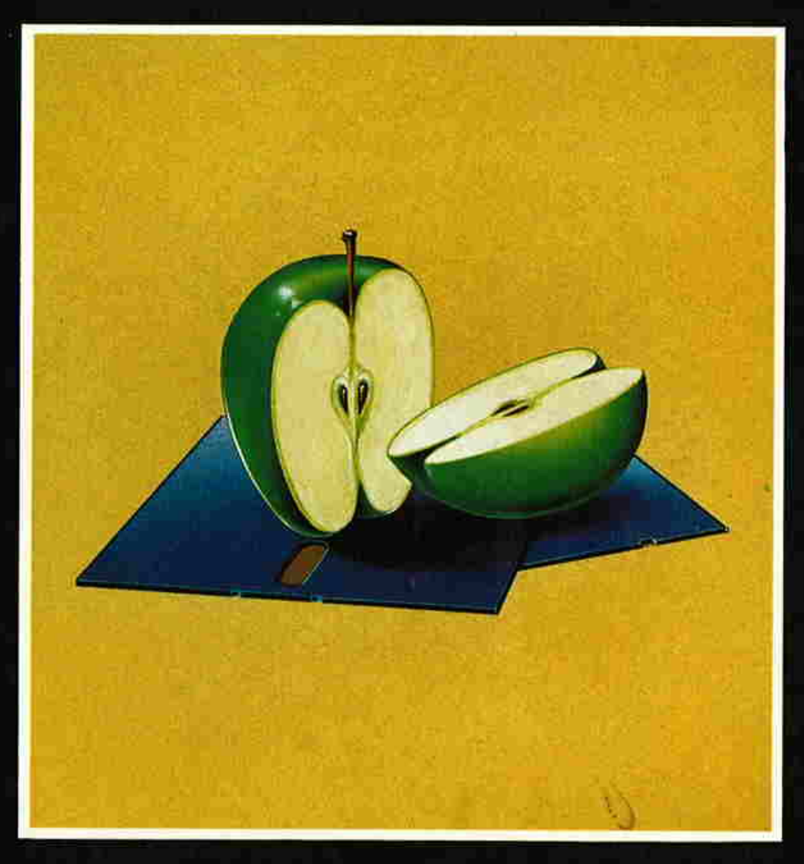
SYBEX

Das ProDOS Handbuch

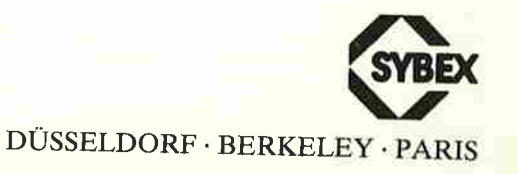


Karen Rice/Timothy Rice



Das ProDOS-Handbuch

T. und K. Rice



DÜSSELDORF · BERKELEY · PARIS

Anmerkungen:
Die Begriffe Apple, Apple II, Apple II+, Apple IIe, Apple IIc, Apple III, Disk II, DOS 3.3, ProDOS, ProFile, Applesoft BASIC, Integer BASIC, The Filer und CONVERT sind Warenzeichen, eingetragene Warenzeichen oder urheberrechtlich geschützt von Apple Computer, Inc.
ThunderClock ist ein Warenzeichen von ThunderWare, Inc.

Originalausgabe in Englisch
Titel der amerikanischen Ausgabe: The ProDOS Handbook

Deutsche Übersetzung: Alfons Wirtz
Umschlaggestaltung: Nicolae Razumieff/tgr
Satz: tgr – typo-grafik-repro gmbh., Remscheid
Gesamtherstellung: Druckerei Hub. Hoch, Düsseldorf

Der Verlag hat alle Sorgfalt walten lassen, um vollständige und akkurate Informationen zu publizieren. SYBEX-Verlag GmbH, Düsseldorf, übernimmt keine Verantwortung für die Nutzung dieser Informationen, auch nicht für die Verletzung von Patent-, Lizenz- und anderen Rechten Dritter, die daraus resultieren.

ISBN 3-88745-617-3
1. Auflage 1985

Printed in Germany

Inhaltsverzeichnis

Vorwort

Dieses Buch beschreibt ProDOS, das neue Disketten-Betriebssystem für die Apple II-Computerfamilie. ProDOS gibt dem Apple II ein neues Gesicht, bietet neue Leistungen und sichert damit den fortwährenden Erfolg der Maschine, die, wie manche sagen, die Revolution der Personal-Computer ausgelöst hat. ProDOS ist mehr als eine neue Version von DOS 3.3, Apples älterem Betriebssystem. ProDOS ist ein kühner Schritt in die Zukunft, der die Zwänge des DOS 3.3 von den Muskeln Ihres Apple entfernt.

ProDOS bietet gegenüber dem älteren DOS 3.3 eine Reihe von Vorteilen. Es kann viel schneller Daten von einer Diskette oder Festplatte lesen oder darauf schreiben als DOS 3.3. ProDOS-Disketten oder -Festplatten können eine fast unbegrenzte Anzahl von Dateien enthalten, während DOS 3.3 nur 105 Dateien handhaben kann. Die maximale Größe einer einzelnen Datei wurde von 140000 Bytes auf 32000000 Bytes erweitert. Die hierarchische Struktur erlaubt eine bessere Kontrolle der Dateien. Fast alle alten DOS 3.3-Befehle sind erhalten geblieben und durch Zusätze verbessert worden. Eine Reihe mächtiger neuer Befehle wurde hinzugefügt.

Dieses Buch ist für jeden gedacht, der mit ProDOS arbeiten möchte, vom Anfänger bis zum erfahrenen Computerbenutzer. Für den Anfänger ist es eine wichtige Hilfe beim Einarbeiten in ProDOS. Kennen Sie sich schon etwas mit ProDOS aus und möchten Sie anfangen, damit zu programmieren, so wird Ihnen dieses Buch ein wertvoller Führer durch die Eigenheiten von ProDOS sein. Der erfahrene Programmierer, der einen schnellen Start und mehr technische Daten über ProDOS sucht, kann die ersten Kapitel überschlagen und sich in die für ihn interessanteren Gebiete vertiefen.

Das Buch ist in zwei Teile gegliedert. Die ersten drei Kapitel behandeln besonders die allgemeineren Aspekte von ProDOS: was es ist, wie es sich von DOS 3.3 unterscheidet und wie man die ProDOS-Dienstprogramme bei einem Apple IIe, einem Apple II+ oder einem Apple IIc benutzt. Im zweiten Teil des Buchs besprechen wir, wie man ProDOS in Programmen

anwendet. Während einiges von dem Stoff, der in diesen Kapiteln behandelt wird, notwendigerweise ziemlich technisch ist, sollte das Buch doch im ganzen für jeden, der sich für ProDOS interessiert, leicht lesbar sein. Wir haben versucht, auf die wenigen Stellen im Text hinzuweisen, die für ein gutes Allgemeinverständnis nicht wichtig sind. Sie können sie überschlagen, wenn Sie nicht an den genaueren technischen Details von ProDOS interessiert sind.

Kapitel 1 beginnt mit einer Besprechung der Werkzeuge, die man für ProDOS braucht: dem Apple II-Computer, dem Diskettenlaufwerk und den Disketten. (Wir behandeln hauptsächlich IIe- und IIc-Computer. ProDOS läuft aber auch auf einem auf 64K erweiterten Apple II+. Die II+-Version von ProDOS ist im wesentlichen mit der IIe-Version identisch.) Wir vergleichen ProDOS mit DOS 3.3, um zu zeigen, wo Verbesserungen gemacht worden sind und warum ProDOS mehr Leistung bietet. Kapitel 2 beschreibt die Dienstprogramme auf der ProDOS User's Disk für den Apple IIe. Wir werden Sie Schritt für Schritt durch jede einzelne verfügbare Funktion führen, so daß Sie sie in sehr kurzer Zeit beherrschen werden. Mit diesen Dienstprogrammen können Sie schon die gebräuchlichsten ProDOS-Operationen durchführen (wie z. B. Disketten kopieren oder deren Inhalt katalogisieren), selbst wenn Sie vorher noch gar nicht zu programmieren gelernt haben. Kapitel 3 behandelt die etwas davon abweichenden Dienstprogramme auf der Diskette mit den System-Dienstprogrammen für den Apple IIc. Hiervon gibt es eine Version in deutscher Sprache. In diesem Kapitel werden auch die Unterschiede zwischen den beiden ProDOS-Versionen besprochen.

In Kapitel 4 beginnt der mehr programmorientierte Abschnitt dieses Buchs. Dieses Kapitel behandelt die ProDOS-Verzeichnisstruktur, und wie man sie dazu benutzt, Dateien auf einer Diskette oder Festplatte zu organisieren; es sollte von allen ProDOS-Anwendern gelesen werden. Der gelegentliche Anwender kann die technischen Abschnitte (auf die im Text hingewiesen wird) überschlagen, sollte das Kapitel aber trotzdem lesen, um ein Gefühl für die Arbeitsweise von ProDOS zu bekommen. Der mehr technisch Interessierte kann die detaillierten Beschreibungen studieren: wie ProDOS Daten aufzeichnet und Dateien organisiert.

Kapitel 5 behandelt die ProDOS-Befehle, die im BASIC-Modus zur Verfügung stehen. DOS 3.3-Anwender werden sich besonders für die neuen ProDOS-Befehle und die Verbesserungen an den alten DOS 3.3-Befehlen interessieren. Hier werden auch die DOS 3.3-Befehle erwähnt, die weggelassen worden sind. Dieses Kapitel sollte von allen ProDOS-Anwendern gelesen werden – jeder, der den Computer häufig benutzt,

möchte gerne Nutzen aus den ProDOS-Funktionen ziehen und sich dazu nicht unbedingt erst durch die ProDOS- Dienstprogramme arbeiten müssen.

In Kapitel 6 wird behandelt, wie unter ProDOS Speicherplatz vergeben wird. Hier wird die System-Bitabbildung beschrieben, mit der ProDOS die Belegung des Speichers aufzeichnet. Außerdem werden die von ProDOS benutzten Speicherstellen angegeben, und die ProDOS-Ladesequenz wird vorgestellt, so daß Sie genau wissen, wie ProDOS in den Speicher geladen wird. Dieses Kapitel wird sowohl dem gelegentlichen Anwender als auch dem empfohlen, der an den tieferen technischen Details von ProDOS interessiert ist.

In Kapitel 7 und 8 werden verwandte Themen besprochen. Beide Kapitel drehen sich um die Behandlung von Dateien durch die ProDOS-Befehle. Kapitel 7 behandelt die Verwendung von ProDOS bei Textdateien, ein Thema, das jedem BASIC-Programmierer am Herzen liegen sollte. In Kapitel 8 gehen wir besonders auf die Behandlung binärer Dateien durch ProDOS ein. Beide Kapitel enthalten Arbeitsbeispiele, die die dort besprochenen ProDOS- Befehle erläutern.

Kapitel 9 beschreibt, wie ProDOS bei Grafikprogrammen und zur Tonerzeugung eingesetzt wird. Hier erfahren Sie, wie man mit Hilfe binärer Dateien Grafikbilder auf einer Diskette abspeichern oder von einer Diskette laden kann, und zwar anhand von Programmbeispielen, die die Technik in Aktion zeigen. Wir besprechen auch, wie man den Speicherbereich der hochauflösenden Grafik vor anwachsenden BASIC-Programmen schützt.

Kapitel 10 behandelt das ProDOS-MLI. Dies ist das Maschinensprache-Interface, das alle Ihre ProDOS-Befehle von BASIC in Maschinensprache übersetzt, so daß sie vom Apple ausgeführt werden können. Dieses Kapitel enthält sehr technischen Stoff, aber jeder Leser sollte diese Seiten wenigstens überfliegen, um die Arbeitsweise von ProDOS besser zu verstehen. Der fortgeschrittene Programmierer wird hier viele Details finden.

Kapitel 11 beschließt das Buch mit einer kurzen Diskussion der ProDOS-Systemprogramme und des Monitors, eines in Ihrem Apple residenten Maschinensprache-Programms, mit dem Sie den Inhalt des Speichers überprüfen und abändern können. Sie können damit auch Maschinensprache-Programme schreiben, die das ProDOS-MLI direkt benutzen.

Wir haben einige Kapitel als Anhang hinzugefügt, damit Sie im Text besprochene Dinge leicht nachschlagen können. Anhang A enthält eine

Liste der ProDOS-Dateitypen. In Anhang B sind die Fehlermeldungen der ProDOS-Dienstprogramme aufgeführt. Anhang C enthält eine Liste der Fehlercodes und Fehlermeldungen, die von ProDOS-Befehlen erzeugt werden können, und Anhang D eine Liste der Fehlercodes des Applesoft-BASIC, die in die gleiche Speicherstelle zurückgegeben werden.

In Anhang E sind die ASCII-Zeichen (American Standard Code for Information Interchange) aufgeführt, die der Apple II benutzt. Anhang F enthält eine Liste der Zeichen, die dazu verwendet werden, bei Applesoft-BASIC reservierte Wörter auf der Diskette abzuspeichern.

In Anhang G finden Sie eine kurze Nachschlagetabelle der MLI- Befehle und deren Parameter. Dahinter folgt in Anhang H eine Liste der Fehlermeldungen, die das MLI in den Akkumulator zurückgibt, mit ihren Bedeutungen. Diese Informationen sind besonders wichtig für den Maschinensprache- oder Assemblerprogrammierer.

Die nächsten drei Kapitel des Anhangs haben alle mit Speicherbelegung zu tun. Anhang I beschreibt, wie sich Applesoft und ProDOS den Bereich der nullten Speicherseite (zero page) teilen. Anhang J enthält einen Speicherplan mit der Aufteilung des gesamten 64K großen Speicherbereichs unter ProDOS. Anhang K schließlich erläutert, wie ProDOS die System-Global-Seite benutzt, um mit dem MLI zu kommunizieren.

Eine nicht modifizierte ProDOS User's Disk läuft nur auf den Originalgeräten der Firma Apple. Das letzte Kapitel des Anhangs enthält eine Beschreibung, wie Sie mit Hilfe des Monitors die ProDOS User's Disk auch auf einem Apple II-kompatiblen Computer zum Laufen bringen können.

Kapitel 1

Einführung in ProDOS

Aufgabe dieses Kapitels ist es, Sie mit den Grundelementen von ProDOS und Ihrem Apple II-Computerssystem vertraut zu machen. Für diejenigen, die noch keine Erfahrung mit Computern haben, geben wir einen kurzen Überblick über die Arbeitsweise eines Computers und die Funktion eines Betriebssystems. Wenn Sie sich mit Computern schon einigermaßen auskennen, können Sie die nächsten Seiten überschlagen und beim Abschnitt mit dem Titel „Überblick über ProDOS" weiterlesen. Dort geben wir eine kurze Zusammenfassung von ProDOS und vergleichen es mit DOS 3.3, dem Vorläufer von ProDOS bei den Apple II-Betriebssystemen. Am Ende dieses Kapitels werden Sie genügend Hintergrundinformationen haben, um mit der Anwendung von ProDOS zu beginnen.

DER APPLE II

Ein typisches Apple IIe-Computersystem besteht aus einem Apple IIe-Computer, zwei Diskettenlaufwerken, einem Monitor und einem Drukker. Ein Apple IIc sieht demgegenüber etwas anders aus, da ein Diskettenlaufwerk schon fester Bestandteil des Computers ist. Dieser Unterschied beeinflußt die Arbeitsweise von ProDOS nicht.

Der Computer

Der Computer ist das schreibmaschinenähnliche Gerät mit der Tastatur. Über diese Tastatur werden Sie im allgemeinen zum Computer „sprechen". Sind Computer und Monitor eingeschaltet, wird jedes Zeichen, das Sie durch ein Tippen auf die Tastatur eingeben, auf dem Bildschirm des Monitors dargestellt.

Wenn Sie die Abdeckung des Apple IIe öffnen und hineinschauen, sehen Sie an der linken Seite einen Metallkasten – das ist das Netzteil. Es ist mit einer deutlichen Warnung beschriftet, den Kasten nicht zu öffnen. In sei-

nem Innern herrschen gefährliche Spannungen, und Sie können einen empfindlichen Stromschlag bekommen und Ihren Computer beschädigen, wenn Sie daran herumbasteln.

Rechts neben dem Netzteil befindet sich ein großes grünes Plastikbrett, auf dem eine Anzahl von schwarzen Objekten befestigt sind. Das sind die Computerchips, über die jeder spricht. Der lange Chip in der Nähe der Brettmitte mit der Nummer 6502 ist der Mikroprozessor. Ein Mikroprozessor ist mit dem Motor in einem Auto vergleichbar – er liefert die Leistung, die alles andere in Bewegung setzt. Hinten im Computer sind acht lange Steckplätze angebracht. Sie können in diese Steckplätze „Karten" hineinstecken und so das System mit Funktionen ausstatten, die sonst nicht zur Verfügung stehen.

Wenn Sie einen Apple IIc besitzen, wird Ihr Gerät etwas anders als der oben beschriebene Apple IIe aussehen. Der Apple IIc ist als tragbarer und kompakter Computer konstruiert. Deshalb sind seine Erweiterungsmöglichkeiten auf die Steckeranschlüsse an seiner Rückseite beschränkt. (Sie sollten übrigens Ihren IIc-Computer nicht öffnen, da sonst das Garantierecht erlischt.)

Im Inneren Ihres Computers gibt es eine Reihe von Chips, die als RAM bezeichnet werden. Diese sind gemeint, wenn über Speicher (Memory) gesprochen wird. In Ihrer Maschine gibt es noch eine weitere Art von Speicher, der ROM genannt wird. Das ROM wird normalerweise nicht zum Arbeitsspeicher gerechnet.

RAM steht für „Random Access Memory" (Speicher für wahlfreien Zugriff) und ROM für „Read Only Memory" (Nur-Lese-Speicher). Der Inhalt des ROM bleibt immer gleich, während sich der Inhalt des RAM ständig ändert. Wenn Sie Ihr System starten, wird ProDOS ins RAM geladen. Auch jedes andere Programm wird von der Diskette ins RAM übertragen, bevor es ausgeführt werden kann. Sie können sich das RAM als eine Tafel vorstellen, die dauernd beschrieben, gelöscht und wieder neu beschrieben wird.

Speicherplatz wird in Bytes gemessen. Ein Byte ist im wesentlichen der Speicherplatz, der benötigt wird, um ein einzelnes Zeichen wie etwa den Buchstaben A zu speichern. Ein Kilobyte hat 1024 Bytes. Der Apple IIe besitzt standardmäßig 64K (K steht für Kilobyte) RAM-Speicherplatz. Der Apple II+ hat 48K RAM; er kann jedoch durch Hinzufügen einer Speicher- oder Language-Karte in Steckplatz (slot) 0 auf 64K erweitert werden. Ein Apple IIc besitzt eine Standard-Speichergröße von 128K. Sie brauchen mindestens 64K RAM in Ihrem Apple, bevor Sie ProDOS von der Diskette in den Speicher laden können.

Laufwerke, Disketten und Festplatten

Ein Laufwerk ist im wesentlichen eine „Kreuzung“ aus Tonbandgerät und Plattenspieler. Wie ein Plattenspieler speichert ein Laufwerk Informationen auf einer dünnen, runden Scheibe. Wie ein Tonbandgerät kann es sowohl neue Informationen abspeichern als auch alte Informationen wiedergeben. Die Prozedur, auf einer Diskette alte Daten zu suchen und in den Speicher zu übertragen, heißt Leseoperation (read). Eine Schreiboperation (write) überträgt neue Daten aus dem Speicher auf eine Diskette.

Normalerweise werden Disketten (floppy disks) benutzt, um Daten und Programme eines Apple zu speichern. Das Standard-Diskettenlaufwerk von Apple (auch Disk II genannt) verwendet einseitige Disketten mit einfacher Dichte und 5 1/4 Inch Durchmesser. Einseitig bedeutet, daß nur auf einer Seite der Diskette Daten gespeichert werden. Einfache Dichte bezieht sich darauf, wie dicht die Daten auf die Diskette gepackt werden.

Eine Diskette besteht aus einer dünnen, von Plastik umhüllten Schicht aus Aufnahmematerial. Das Loch in der Mitte wird zum Festhalten und Drehen der Diskette im Laufwerk benutzt. Der längliche Schlitz in der Nähe des Mittellochs gibt die Oberfläche der Diskette frei, so daß sie vom Laufwerk gelesen werden kann. Auf der linken Seite der Diskette befindet sich eine Schreibschutzkerbe. Wenn diese Kerbe mit einem Stück Aluminiumpapier verdeckt ist, kann man weder neue Dateien auf die Diskette schreiben noch bereits existierende Dateien verändern. Bevor Sie neue Daten auf die Diskette schreiben können, müssen Sie das Papier wieder entfernen. Diese Kerbe macht es möglich, wichtige Daten vor einer zufälligen Beschädigung zu schützen.

Stellen Sie sich eine Diskette als ein dünnes, zerbrechliches Aufnahmealbum vor. Falls Sie die freiliegende Oberfläche der Diskette berühren, die Diskette verbiegen oder sie auf eine andere Art beschädigen, können alle Informationen oder Programme auf der Diskette verloren sein.

Wenn Sie eine Diskette benutzen wollen, legen Sie sie ins Laufwerk, indem Sie die Laufwerkstür öffnen und die Diskette vorsichtig hineinschieben. Wenden Sie keine Gewalt an, wenn die Diskette nicht ganz hineingeht. Nehmen Sie sie heraus, und versuchen Sie noch einmal, die Diskette hineinzuschieben. Haben Sie dann immer noch Probleme, so sehen Sie nach, ob irgend ein Hindernis entfernt werden muß. Denken Sie daran, die Tür zu schließen, wenn sich die Diskette im Laufwerk befindet.

Apple verkauft zwei Arten von Laufwerken für den Apple II. Das übliche ist das Diskettenlaufwerk, das auch Disk II genannt wird. Dann gibt es

noch die sogenannte ProFile. Hierbei handelt es sich um ein Festplattenlaufwerk (hard disk). Festplattenlaufwerke sind geschlossene Einheiten. Bei einem Festplattenlaufwerk können Sie nicht wie bei einem Diskettengerät die Platte herausnehmen. Festplatten haben den Vorteil, daß ihre Speicherkapazität viel größer und ihre Zugriffszeit viel kürzer als bei einem Diskettenlaufwerk ist. Eine ProFile kann zum Beispiel 5 Millionen Bytes Daten speichern, während eine Diskette für den Apple II nur eine Speicherkapazität von 140000 Bytes hat. (Bis jetzt kann die ProFile noch nicht an einen Apple IIc angeschlossen werden.)

Ein Laufwerk ist mit mit dem Apple IIe über ein flaches Bandkabel verbunden. Dieses Kabel führt durch eine der Öffnungen auf der Rückseite des Gerätes in den Computer und ist dort an eine der Karten in den Erweiterungssteckplätzen angeschlossen. Diese Karte heißt Laufwerkkontrollkarte (disk controller card). Bei einem Apple IIc ist die Verbindung zwischen dem eingebauten Laufwerk und der Hardware des Computers nicht sichtbar, außer wenn Sie das Gerät öffnen (was nicht ratsam ist).

Eine Laufwerkkontrollkarte für Diskettengeräte hat zwei Steckkontakte, die deutlich mit Drive (Laufwerk) 1 und Drive 2 markiert sind. Wenn Sie nur ein Laufwerk haben, muß sie an den mit Drive 1 bezeichneten Steckkontakt angeschlossen werden. An eine solche Karte können zwei Diskettenlaufwerke angeschlossen werden. Wenn Sie mehr als zwei Laufwerke haben, brauchen Sie eine weitere Laufwerkkontrollkarte. Die erste Laufwerkkontrollkarte wird normalerweise in Steckplatz 6 gesteckt, und eine zusätzliche kommt in Steckplatz 5 oder 4. An eine Interface-Karte für ein Festplattenlaufwerk kann nur ein Laufwerk angeschlossen werden.

Im Verlauf dieses Buchs werden wir oft auf Disketten oder Laufwerke zu sprechen kommen. In den meisten Fällen wird alles, was wir über Diskettenlaufwerke sagen werden, für Festplattenlaufwerke praktisch genauso zutreffen. Wenn es einen Unterschied gibt, werden wir besonders darauf hinweisen.

Der Monitor

Ein Monitor ist ein Gerät, das wie ein Fernseher aussieht. Er wirkt wie ein Fenster in den Computer. Wenn Sie einen Befehl auf der Tastatur tippen, gibt der Apple ihn auf dem Bildschirm des Monitors wieder. Auch alle Meldungen eines Programms, das gerade läuft, erscheinen dort. Obwohl es einen Adapter gibt, mit dem man auch ein Fernsehgerät als Monitor benutzen kann, ziehen die meisten Leute einen speziell für diese Aufgabe bestimmten Monitor vor, weil er ein besseres Bild erzeugt.

Der Drucker

Ihr Computer benutzt einen Drucker, um Daten zu Papier zu bringen. Sie können Daten fast jeden Typs ausdrucken. Dazu gehören Briefe, Finanzberichte und Inventarlisten, die Sie mit einem Programm erzeugt haben, Aufzeichnungen (listings) von Programmen, die Sie geschrieben haben, und spezielle Graphen oder Karten, die Sie mit einem Grafikprogramm hergestellt haben. Bei Personal-Computern wie dem Apple II gibt es normalerweise zwei Typen von Druckern. Der eine und bei weitem populärste ist der Matrixdrucker. Der andere ist der Typenraddrucker. Ein Matrixdrucker erzeugt Zeichen mit kleinen Punkten aus Tinte. Matrixdrucker sind im allgemeinen billiger und schneller als Typenraddrucker. Trotzdem werden sie manchmal für bestimmte Aufgaben als nicht ausreichend empfunden, weil sie die Zeichen nicht so klar drucken wie ein Typenraddrucker.

Typenraddrucker drucken genauso wie eine Schreibmaschine. Sie besitzen ein Rad oder ein fingerhutförmiges Gebilde, das alle Zeichen enthält, die sich auf der Tastatur befinden. Wenn der Drucker arbeitet, befindet sich das Rad in ständiger Drehung, damit die richtige Type gegen das Farbband geschlagen wird. Das geht viel langsamer vor sich als das Drukken mit dem Matrixdrucker. Typenraddrucker sind erheblich teurer als Matrixdrucker, und sie können nicht, wie viele Matrixdrucker, Grafiken und Bilder drucken.

Ihr Drucker ist normalerweise über eine Interface-Karte in Steckplatz 1 mit Ihrem Apple verbunden. Es gibt zwei Haupttypen von Interface-Karten: das serielle und das parallele Interface. Diese Bezeichnungen beziehen sich auf die Art und Weise, wie die Daten von Ihrem Computer zum Drucker übertragen werden. Bei der seriellen Übertragungsweise werden die Daten einzeln hintereinander übertragen, bei der parallelen Übertragungsweise hingegen werden mehrere Daten gleichzeitig übertragen. Sie können sich den Unterschied wie den zwischen einer einspurigen Straße und einer Autobahn vorstellen.

(Bei einem Apple IIc-Computer können Sie nur einen seriellen Drucker verwenden, weil er keinen parallelen Anschluß hat. Sie brauchen den Drucker nur am Ausgang für den seriellen Drucker auf der Rückseite des Computers anzuschließen.)

Es ist wichtig zu wissen, daß das serielle und das parallele Interface nicht zusammenarbeiten können. Wenn Sie einen seriellen Drucker haben, brauchen Sie ein serielles Interface. Haben Sie einen parallelen Drucker, benötigen Sie ein paralleles Interface.

WIE COMPUTER ARBEITEN

Bevor der Computer irgend etwas machen kann, muß er programmiert werden. Ein Programm ist eine Folge von Instruktionen, die der Computer in einer bestimmten Reihenfolge ausführt. Wenn ein Programm fertiggestellt ist, wird es normalerweise als Datei auf einer Diskette gespeichert.

Im Innern des Computers ist der Mikroprozessor der Chef. Er liest jede einzelne Instruktion und tut alles, was nötig ist, damit sie ausgeführt werden kann.

Die Laufwerke können mit einer Registratur verglichen werden. Die Disketten, die Sie in Ihren Laufwerken benutzen, entsprechen dann den Aktenschränken in der Registratur. Jeder einzelne von ihnen kann viele verschiedene Akten enthalten.

Wenn Sie dem Computer sagen, er soll ein Programm, das auf einer Diskette gespeichert ist, laufen lassen, muß er es zuerst in den Speicher laden. Der Speicher dient als Schreibtisch oder Arbeitsplatz des Mikroprozessors. Sobald sich das Programm im Speicher befindet, liest der Prozessor alle Anweisungen und führt sie der Reihe nach aus.

Wenn das Programm weitere Daten von der Diskette braucht, wie zum Beispiel Namen und Adressen oder einen Geldbetrag, den man Ihnen schuldet, dann wird der Prozessor – gemäß den Anweisungen des Programms – auf der Diskette nach ihnen suchen. Das Programm liest die Daten in den Speicher, damit sie vom Prozessor verarbeitet werden können. Wenn das Programm diese Daten ändern soll, schreibt der Prozessor die neuen Daten auf die Diskette. Wenn es keine Änderungen gibt, brauchen keine Daten auf die Diskette zurückgeschrieben zu werden – die Originaldatei befindet sich ja noch auf der Diskette, in den Speicher ist nur eine Kopie der Daten übertragen worden.

Nach der Ausführung des Programms bleibt das Programm im Speicher liegen. Wenn Sie ein anderes Programm laufen lassen, wird das erste Programm aus dem Speicher entfernt und geht verloren, falls Sie es nicht vorher auf einer Diskette abgespeichert haben. Das Programm geht auch verloren, wenn Sie den Computer ausschalten.

BETRIEBSSYSTEME

Ein Betriebssystem ist ein Satz von Regeln und Ausführungsbestimmungen, nach denen der Computer bei der Durchführung Ihrer Anweisungen vorgeht. Wenn Sie den Apple einschalten, ist immer ein einfaches

Betriebssystem vorhanden, das Ihre Befehle interpretiert und dem Prozessor sagt, was Sie von ihm wollen. Darüber hinaus kann Ihr Apple ein weiteres Betriebssystem für Operationen, die mit Disketten zu tun haben, benutzen. Dieses Betriebsystem muß von der Diskette in den Speicher geladen werden, bevor es verwendet werden kann.

ÜBERBLICK ÜBER ProDOS

ProDOS ist ein neues Disketten-Betriebssystem, das entworfen wurde, um die Leistung des Apple II zu maximieren. ProDOS ist schneller, leichter zu handhaben und mächtiger als das alte Betriebssystem DOS.

Wenn Sie Ihren Apple II vor 1984 gekauft haben, haben Sie DOS 3.3 als Betriebssystem zu Ihrem Disk II-Laufwerk erhalten und müssen ProDOS getrennt kaufen. Haben Sie Ihren Apple IIe nach 1984 gekauft, ist Ihnen ProDOS in Form der ProDOS User's Disk (ProDOS-Anwenderdiskette) mitgeliefert worden. Sie bekommen ProDOS auch beim Kauf eines Apple IIc, und zwar in Form der Diskette mit den System-Dienstpogrammen in deutscher Fassung. Die Menüstruktur der System-Dienstprogramme bei diesen beiden ProDOS-Versionen ist ziemlich unterschiedlich.

ProDOS (die Buchstaben stehen für Professional Disk Operating System) handhabt alle Operationen, die mit dem Speichern von Daten auf einer Diskette oder dem Zurückholen von Daten von der Diskette in den Speicher zu tun haben. Das Betriebssystem handelt wie ein Verkehrspolizist, der den Fluß der Daten nach Ihren Befehlen dirigiert. Es tut nichts, ohne daß es von Ihnen oder von einem gerade laufenden Programm dazu aufgefordert wurde.

ProDOS ist ein Betriebssystem, das leicht zu handhaben ist. Bei den meisten Mikrocomputern müssen Sie alle Befehle auswendig kennen, wenn Sie das System benutzen wollen. Bei ProDOS werden Ihnen Auswahlmöglichkeiten in Form eines Menüs vorgestellt, und Sie treffen einfach Ihre Wahl. Wenn das Betriebssystem zusätzliche Angaben braucht, fragt es Sie, was es von Ihnen wissen möchte. Drücken Sie versehentlich die falsche Taste, fragt es Sie nochmals nach der korrekten Eingabe. Bei ProDOS braucht man sich nicht an komplizierte Befehle zu erinnern.

Wenn Sie Ihren Apple einschalten, bietet Ihnen ProDOS ein Menü von Auswahlmöglichkeiten an. Wenn Sie Fragen haben, wie es weitergeht, finden Sie im allgemeinen die gesuchte Antwort, indem Sie einfach die ?-Taste drücken. (Beim IIc müssen Sie die Offene-Apfel- und ?-Taste gleichzeitig drücken, wenn Sie Hilfe haben möchten.) Das bringt den

sogenannten Tutor auf den Bildschirm. Der Tutor erklärt Ihnen die Operationen, mit denen Sie sich gerade beschäftigen.

ProDOS wird sich Ihnen auf einem Apple IIc in einer etwas anderen Aufmachung vorstellen als auf einem Apple IIe oder II+, aber die Betriebssysteme selbst sind gleich. Das heißt, daß alle Befehle, die Ihnen im BASIC oder im MLI zur Verfügung stehen, gleich funktionieren und die gleichen Ergebnisse liefern. Die Unterschiede liegen in der Menüstruktur und in den Dienstprogrammen. In den nächsten beiden Kapiteln dieses Buchs besprechen wir die System-Dienstprogramme von ProDOS für den IIe und den IIc, wobei die ProDOS User's Disk für den IIe in Kapitel 2 und die deutschsprachige Dienstprogramm-Diskette für den IIc in Kapitel 3 behandelt werden. Dort werden wir auch die Unterschiede zwischen den ProDOS-Versionen hervorheben. (Weil vieles in den Kapiteln 2 und 3 gleich ist, schlagen wir vor, daß Sie Kapitel 2 lesen und dann nach Kapitel 4 springen, wenn Sie einen IIe besitzen. Haben Sie einen IIc, dann können Sie Kapitel 3 anstelle von Kapitel 2 lesen.) Das in den folgenden Paragraphen beschriebene Hauptmenü betrifft die IIe-Version von ProDOS.

Einer der wesentlichsten Bestandteile von ProDOS sind die Datei-Dienstprogramme (filer) im Hauptmenü. Mit Hilfe dieser Datei-Dienstprogramme können Sie mit einem Minimum an Aufwand und Verdruß einzelne Dateien oder ganze Disketten kopieren oder wieder löschen, wenn Sie sie nicht mehr benötigen. Sie helfen Ihnen auch, die Dateien auf einer Diskette vernünftig zu organisieren. (Diese Programme gibt es natürlich auch in der IIc- Version von ProDOS.)

Aus Gründen der Kontinuität zwischen DOS und ProDOS hat Apple dafür gesorgt, daß Dateien von einem System ins andere konvertiert werden können. Die meisten Applesoft-BASIC-Programme verhalten sich unter beiden Betriebssystemen gleich. Diese Aufwärtsmobilität von DOS nach ProDOS ist eine wertvolle und wichtige Eigenschaft von Apples neuem Betriebssystem. Programme, die direkte Eingriffe ins DOS machen, werden jedoch unter ProDOS möglicherweise nicht laufen. Deshalb können Sie kopiergeschützte Programme unter Umständen nicht von DOS nach ProDOS übertragen.

ProDOS bietet in seinem Hauptmenü drei weitere Optionen an. Wenn Sie Display Slot Assignments wählen, wird ProDOS Ihnen genau sagen, was sich in den einzelnen Erweiterungssteckplätzen Ihres Apple II befindet. Es teilt Ihnen den Namen der Diskette mit, mit der Sie das System gestartet haben, und wieviel Speicherplatz Ihr Apple II hat. Wenn Sie Display/Set Time wählen, können Sie Datum und Uhrzeit einstellen und allen Ihren Dateien das Datum ihrer Herstellung oder ihrer letzten Ände-

rung aufstempeln lassen. Haben Sie eine Uhr/Kalender-Karte in Ihrem Apple, dann überprüft ProDOS Datum und Uhrzeit automatisch, so daß diese Option überflüssig wird, es sei denn, Sie möchten aus einem bestimmten Grund ein anderes Datum einsetzen.

Die letzte Option im Hauptmenü von ProDOS ist Applesoft-BASIC. Wenn Sie diese Option wählen, können Sie einzelne ProDOS-Befehle direkt unter Umgehung der Menüstruktur anwenden. Hier können Sie auch eigene BASIC-Programme erstellen, in denen ProDOS-Befehle benutzt werden.

ProDOS IM VERGLEICH ZU DOS

DOS 3.3, Apples vorheriges Betriebssystem für Apple II-Computer, stand die größte Fülle an Software von allen Betriebssystemen in der Welt der Personal-Computer zur Verfügung. Millionen waren mit DOS vertraut und haben es gerne benutzt. Bei DOS 3.3 gab es jedoch einige ernsthafte Einschränkungen, die Apple glaubte entfernen zu müssen, wenn der Apple II weiterhin ein starkes Produkt bleiben sollte.

Das größte Problem von DOS 3.3 sind seine beschränkten Möglichkeiten, Platz auf einer Diskette zu nutzen. DOS 3.3 und ProDOS behandeln beide eine Diskette als Datenträger, der gefüllt werden kann. Das größte zugelassene Datenträgerformat unter DOS ist das Format einer einzelnen Diskette. Deshalb muß man unter DOS eine 5 Megabyte große Festplatte in ungefähr 40 Datenträger aufteilen, wenn man die gesamte Speicherkapazität ausnutzen will. Das ist lästig, verwirrend und zeitraubend.

ProDOS ist dafür vorgesehen, Laufwerke mit großer Kapazität zu unterstützen. Unter DOS darf eine Datei höchstens 140 Kilobytes groß sein. Unter ProDOS beträgt die größtmögliche Datei 16 Megabytes, sie ist also mehr als hundertmal größer. Das eröffnet eine neue Welt für den Apple II.

Ein weiteres größeres Problem von DOS ist seine geringe Geschwindigkeit. Nach Angaben von Apple überträgt DOS 3.3 Daten vom und zum Diskettenlaufwerk mit ungefähr einem Kilobyte pro Sekunde. Das hört sich nach viel an, aber größere und schnellere Laufwerke können Daten viel schneller verarbeiten. ProDOS kann Daten zum oder vom Laufwerk mit acht Kilobytes pro Sekunde übertragen, ungefähr achtmal so schnell wie DOS. (Unter typischen Bedingungen schrumpft dieser theoretische Vorteil leicht auf ungefähr fünf zu eins zugunsten von ProDOS.)

Unter DOS ist ein Datenträger auf 105 Dateien begrenzt. Obwohl diese Zahl ziemlich groß aussieht, wird sie doch schnell inadäquat, wenn man es

mit Laufwerken großer Kapazität, wie etwa mit der ProFile, zu tun hat. Je größer die Kapazität eines Laufwerks, desto wichtiger ist es, daß die Dateien gut organisiert werden können. ProDOS benutzt einen hierarchischen Ansatz zur Strukturierung des Dateisystems. Das bedeutet, daß es Inhaltsverzeichnisse (directories) und Unterverzeichnisse benutzt, um Dateien auf einer Diskette oder einer Festplatte zu organisieren. Jedes Inhaltsverzeichnis kann eine Reihe von Dateien enthalten, von denen einige oder alle wieder Verzeichnisdateien sind, die wieder andere Dateien enthalten. Bei praktischen Anwendungen wird es Ihnen so vorkommen, als ob Sie unter ProDOS eine unbegrenzte Anzahl von Dateien anlegen können. Der Hauptvorteil eines hierarchischen Systems ist, daß es die Dateien besser geordnet und schneller und leichter zugreifbar hält.

Die hierarchische Struktur mag nicht wichtig erscheinen, wenn man es nur mit einer kleinen Anzahl von Dateien auf jeder einzelnen Diskette zu tun hat. Denken Sie daran, Unterverzeichnisse anzulegen, wenn die Anzahl der Dateien auf einer Diskette größer ist als die Anzahl der Zeilen auf einer Bildschirmseite. Dann können Sie Ihre Dateien in Kategorien aufteilen und sie so stärker strukturieren.

ProDOS unterstützt im Gegensatz zu DOS 3.3 die Möglichkeit von Unterbrechungen (Interrupts). Das ist sehr wichtig, wenn Computer in einem Rechnerverbund zusammengeschlossen sind.

ProDOS ist zu den meisten DOS-Programmen kompatibel. Das bedeutet, daß man Programm- und Datendateien von DOS nach ProDOS übertragen kann, um sich der größeren Möglichkeiten des neuen Betriebssystems zu erfreuen. ProDOS ist auch kompatibel zu SOS, dem Betriebssystem des Apple III.

ProDOS ist für Erstanwender leichter zu erlernen als DOS. Das liegt daran, daß die meisten Funktionen, die man braucht, über das Menü angeboten werden, das beim Einschalten des Systems erscheint. Sie brauchen sich die Funktionen nicht zu merken; Sie wählen einfach aus den angebotenen Möglichkeiten aus. Zusätzlich gibt es bei ProDOS den Tutor, der Ihnen direkt auf dem Bildschirm Hilfe anbietet.

Mit ProDOS können Sie außerdem die Möglichkeiten Ihres Apple II besser ausnutzen. Wenn Sie eine Uhr/Kalender-Karte haben, liest ProDOS sie ab, um die richtige Uhrzeit und das richtige Datum zu erhalten. ProDOS benutzt auch den über die 80-Zeichen-Karte zur Verfügung gestellten zusätzlichen Speicherplatz und verbessert dadurch die Ausführungsgeschwindigkeit von Programmen, die sonst intensiver auf Disketten zugreifen würden.

Es liegt in der Natur der Sache, daß Apple einige Nachteile in Kauf nehmen mußte, um alle diese Vorteile zu erreichen. Wenn Sie nur ein Diskettenlaufwerk haben, besitzt DOS 3.3 gewisse Vorzüge. ProDOS erwartet, daß bestimmte Dateien auch nach einem Programmwechsel noch auf der Diskette zu erreichen sind. Das verringert den Speicherfreiraum auf jeder einzelnen Diskette. (Obwohl Sie die Disketten beim Austauschen von Programmen im Laufwerk hin und her wechseln können, wird das doch bald zu einem lästigen Problem bei einem System mit nur einem Laufwerk.) ProDOS belegt auch mehr Speicherplatz als DOS und reduziert damit den Bereich, der für die anderen Programme übrigbleibt. Aus technischen Gründen, die wir später in diesem Buch erklären werden, laufen Programme in Applesoft-BASIC unter ProDOS geringfügig langsamer als unter DOS.

Alles in allem ist ProDOS für den Apple II ein großer Schritt nach vorn. Es vergrößert die Fähigkeiten des Computers auf den wichtigen Gebieten Speicherkapazität, Geschwindigkeit und Rechnerverbund beträchtlich. Wegen seiner Kompatiblität mit der Vergangenheit kann das neue Betriebssystem die meiste für DOS erhältliche Software direkt benutzen. Die Anstrengungen von Apple zur Entwicklung dieses neuen Betriebssystems lassen für die Zukunft eine fortwährende Unterstützung des Apple II erwarten.

Kapitel 2

Die Dienstprogramme auf der ProDOS User's Disk

In diesem Kapitel erklären wir Ihnen die Standardoptionen, die Ihnen über das ProDOS-Menü auf der ProDOS User's Disk (ProDOS- Anwenderdiskette) zur Verfügung gestellt werden. Diese Optionen stehen für Dienstprogramme, die Sie benutzen können. Ein „Dienstprogramm" im Computerjargon ist ein Programm, das bestimmte oft in Anspruch genommene Aufgaben für das Betriebssystem erledigt. Die ProDOS User's Disk enthält folgende Dienstprogramme:

- Tutor
- Datei-Dienstprogramme (Filer)
- Umwandlung von DOS nach ProDOS (DOS to ProDOS Conversion)
- Zeige Steckplatzzuweisungen (Display Slot Assignments)
- Zeige/Setze Uhrzeit (Display/Set Time)
- Applesoft-BASIC

Dieses Kapitel handelt speziell von ProDOS für den IIe; die IIc-Version wird im nächsten Kapitel beschrieben. Das ProDOS- Betriebssystem ist in beiden Versionen gleich, sie unterscheiden sich aber im Menü der Dienstprogramme. Obwohl es sich bei der ProDOS User's Disk nicht um die IIc-Version von ProDOS handelt, läuft sie doch fast genau wie in diesem Kapitel beschrieben auch auf dem IIc. Unterschiede wegen der ungleichen Hardware werden am Ende von Kapitel 3 erwähnt.

AUSDRÜCKE UND DEFINITIONEN

Um mit ProDOS richtig umgehen zu können, müssen wir zuerst einige Ausdrücke definieren. Bei ProDOS werden diese Ausdrücke dazu verwendet, angeschlossene Geräte und Dateien zu finden, wenn sie benutzt

werden sollen. Wenn Sie sie nicht richtig anwenden, kann zweierlei passieren. Wahrscheinlich wird ProDOS Ihren Befehl nicht verstehen und ihn deshalb nicht ausführen. ProDOS kann aber auch Ihren Befehl falsch verstehen und ihn trotzdem ausführen. Im Extremfall kann das dazu führen, daß eine falsche Datei gelöscht wird.

Bevor Sie zum nächsten Abschnitt übergehen, vergewissern Sie sich, daß Sie die folgenden Ausdrücke und Definitionen vollständig beherrschen – es macht den Lernprozeß und den Umgang mit dem System viel leichter.

Datenträger (Volumes)

Von ProDOS aus gesehen ist ein Datenträger eine Diskette oder eine Festplatte wie etwa die ProFile. Jeder Datenträger hat einen Namen, der ihm während des in diesem Kapitel beschriebenen Formatierungsvorgangs zugewiesen wird. ProDOS benutzt diese Namen, um die Disketten, die sich in den Laufwerken befinden, zu identifizieren und auf ihnen etwas zu suchen, wenn Sie einen entsprechenden Befehl eingegeben haben. ProDOS sucht eine Diskette über ihren Namen und nicht über die Nummer des Laufwerks, in dem sie sich befindet. Deshalb ist es wichtig, daß Sie sich mit den Namen von Datenträgern und deren Anwendung vertraut machen.

Nehmen wir zum Beispiel an, Sie haben vier Laufwerke angeschlossen. Wenn sich in allen Laufwerken eine Diskette befindet, können Sie mit ProDOS zwischen vier Datenträgern wählen. Haben alle verschiedene Namen (PRODOS1, PRODOS2, PRODOS3 und PRODOS4), dann hat ProDOS keine Probleme, die gewünschte Diskette zu finden, wenn Sie den Datenträgernamen bei einem der nachfolgenden Dienstprogramme angeben. Falls aber alle vier Disketten den gleichen Namen haben, ist ProDOS nicht in der Lage, nur über den Datenträgernamen festzustellen, welche Sie haben wollen. Sie müssen weitere Informationen angeben (wie etwa Steckplatz und Laufwerk), um ProDOS mitzuteilen, wo es den gewünschten Datenträger suchen soll. Sonst wird ProDOS den ersten nehmen, der auf seinem Weg liegt. ProDOS nimmt den Datenträger von dort, wo die Voreinstellungen für Steckplatz und Laufwerk (die auf den nächsten Seiten besprochen werden) hinzeigen. Diese Voreinstellungen können mit den in diesem Kapitel behandelten Dienstprogrammen vorgenommen und jederzeit wieder geändert werden.

Steckplätze (Slots)

ProDOS verwendet Steckplatznummern ähnlich, wie die Post Namen von Ländern verwendet. Wenn ProDOS auf ein Laufwerk zugreifen soll, muß

es wissen, wo sich das Laufwerk befindet. Indem Sie eine Steckplatznummer angeben, teilen Sie ProDOS eine Stelle innerhalb des Computers mit, wo es suchen kann. Normalerweise sind die Diskettenlaufwerke an eine Laufwerkkontrollkarte in Steckplatz 6 angeschlossen. Wenn Sie mehr als zwei Laufwerke haben, brauchen Sie eine zweite Laufwerkkontrollkarte, die möglicherweise in Steckplatz 5 steckt.

Laufwerknummern

Jedes Laufwerk Ihres Apple IIe ist an eine Laufwerkkontrollkarte in einem der Erweiterungssteckplätze 6 oder 5 im Innern des Computers angschlossen. Eine Laufwerkkontrollkarte hat zwei Anschlüsse, die deutlich mit Drive (Laufwerk) 1 und Drive 2 markiert sind. Sie können sich diese Anschlüsse als Städtenamen Ihrer Laufwerke vorstellen. Das ist etwas anschaulicher als eine Steckplatznummer. Das Laufwerk, das sich an dem mit Drive 1 markierten Anschluß befindet, wird bei ProDOS mit Laufwerk 1 bezeichnet. Wenn Sie die Verbindungen vertauschen würden, so daß das alte Laufwerk 1 jetzt Laufwerk 2 wäre, würde ProDOS das erst merken, wenn Sie versuchten, von einem Laufwerk zu lesen oder darauf zu schreiben.

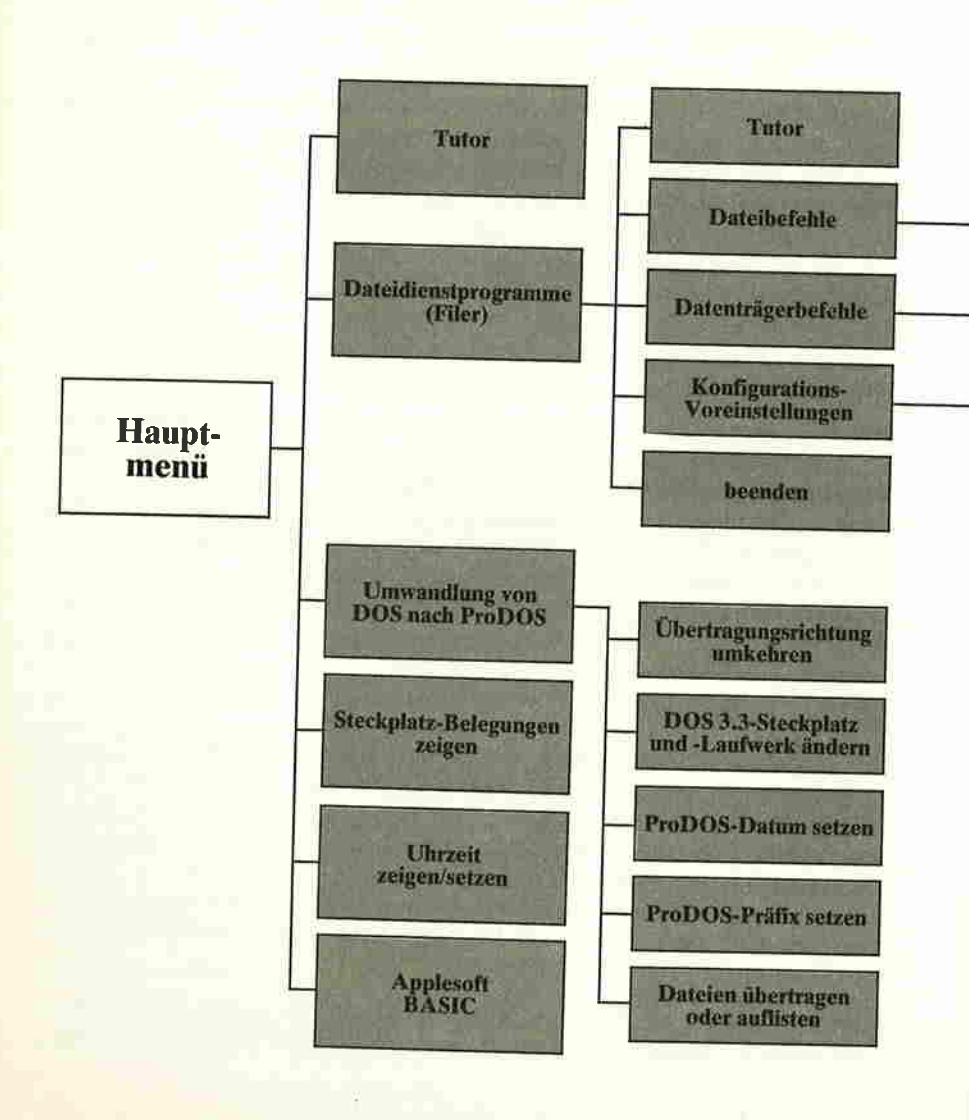
Seien Sie sehr vorsichtig, wenn Sie Laufwerkanschlüsse umstöpseln müssen. Der Motor eines Diskettenlaufwerks wird über den Anschluß an der Laufwerkkontrollkarte mit Strom versorgt. Wenn Sie den Stecker falsch herum oder nicht exakt einstöpseln und dieser Antriebsstrom in eine falsche Ader gerät, wird ein Chip im Laufwerk oder im Computer durchbrennen. Schalten Sie grundsätzlich den Computer aus, und ziehen Sie den Netzstecker, bevor Sie Laufwerkanschlüsse ändern oder eine Steckkarte einsetzen.

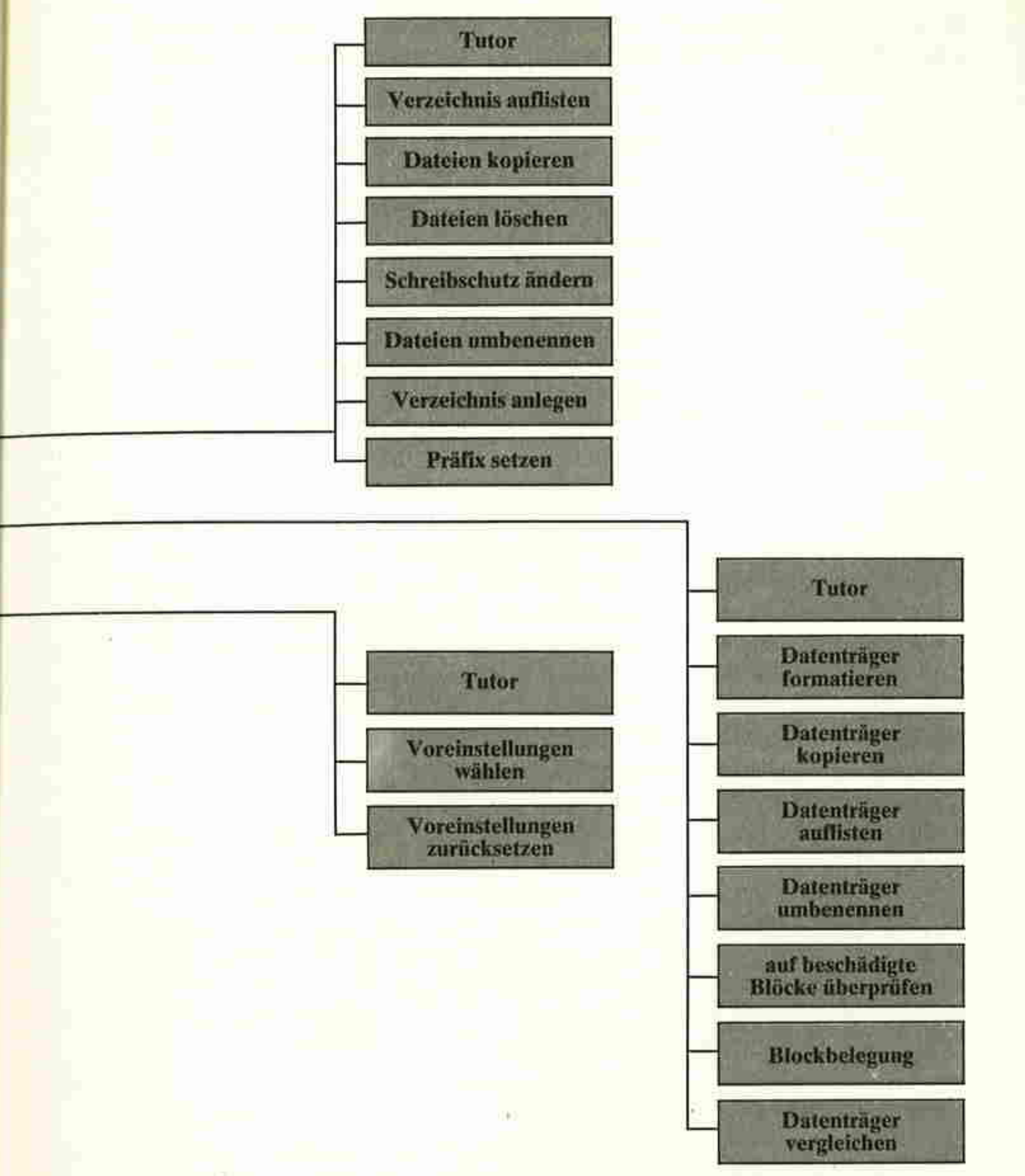
Pfadnamen (Path Names)

Pfadnamen sagen ProDOS, welche Route es einhalten soll, um die von Ihnen gewünschten Dateien zu finden. Ein Pfadname beginnt immer mit dem Zeichen / und einem Datenträgernamen (Diskettennamen). Unserer Analogie folgend, können Sie sich den Namen des Datenträgers als Straßennamen vorstellen. Der übrige Teil des Pfadnamens besteht aus einer Folge von Verzeichnisnamen, die durch das Zeichen / voneinander getrennt sind. Den letzten Teil der Pfadnamens bildet immer der Name der Datei, auf die Sie zugreifen wollen.

Nehmen wir an, Sie möchten zum Beispiel eine Datei mit dem Namen MEINEDATEI auf einem Datenträger mit Namen PRODOS kopieren.

Die Dienstprogramme der ProDOS User's Disk




Tutor
Verzeichnis auflisten
Dateien kopieren
Dateien löschen
Schreibschutz ändern
Dateien umbenennen
Verzeichnis anlegen
Präfix setzen
Tutor
Datenträger formatieren
Datenträger kopieren
Datenträger auflisten
Datenträger umbenennen
auf beschädigte Blöcke überprüfen
Blockbelegung
Datenträger vergleichen
Tutor
Voreinstellungen wählen
Voreinstellungen zurücksetzen

Diese Datei liegt innerhalb einer anderen Datei (einer Verzeichnisdatei) mit dem Namen UNSEREDATEI. Sie müssen einen Pfadnamen bilden, um ProDOS mitzuteilen, wie es MEINEDATEI finden kann. In diesem Fall hieße der Pfadname /PRODOS/UNSEREDATEI/MEINEDATEI. Der Pfadname kann viel länger sein, wenn MEINEDATEI innerhalb einer Folge von Unterverzeichnissen von UNSEREDATEI liegt. Wenn MEINEDATEI im Haupt- oder Wurzelverzeichnis liegt, heißt der Pfadname einfach /PRODOS/MEINEDATEI.

Mit den Namen des Datenträgers, der Verzeichnisse und der tatsächlichen Datei teilen Sie ProDOS die jeweilige Richtung mit, um die gewünschte Datei zu finden. ProDOS folgt diesem Weg in ähnlicher Weise, wie jemand einer Beschreibung folgt, um zu Ihrem Haus zu kommen. Wenn Ihre Richtungsangaben stimmen, wird ProDOS finden, was Sie wünschen; sind sie falsch, kommt ProDOS zurück und fragt noch einmal.

Präfixe

Pfadnamen haben ihre Vorzüge, aber je länger sie sind, desto anfälliger sind sie gegenüber Fehlern. Außerdem ist es lästig, jedesmal einen langen Pfadnamen einzutippen. Deshalb erlaubt Ihnen ProDOS, ein Präfix zu speichern, das an alle Ihre Befehle angefügt wird. Da Sie es nur einmal eintippen, werden Sie sich bei einem Pfadnamen nicht so oft verschreiben. Weil es automatisch vor jeden Pfadnamen gestellt wird, den Sie angeben, erspart es Ihnen eine Menge Tipparbeit.

Nehmen wir zum Beispiel an, Sie möchten MEINEDATEI aus dem Verzeichnis UNSEREDATEI auf dem Datenträger PRODOS löschen. Wenn das Präfix auf /PRODOS/UNSEREDATEI gesetzt ist, brauchen Sie nur den Dateinamen MEINEDATEI im Löschbefehl anzugeben. ProDOS kombiniert automatisch das Präfix mit dem Dateinamen, den Sie angeben, zu einem Pfadnamen und lokalisiert die richtige Datei. Wenn kein Präfix gesetzt ist oder das Präfix auf einen Datenträger oder ein Verzeichnis zeigt, das von /PRODOS/UNSEREDATEI verschieden ist, müssen Sie den ganzen Pfadnamen /PRODOS/UNSEREDATEI/MEINEDATEI eingeben, um die richtige Datei zu löschen.

Joker-Zeichen (Wild Cards)

In einem Pokerspiel kann ein Joker (wild card) benutzt werden, um jede beliebige andere Karte darzustellen. Ein Joker-Zeichen bei ProDOS ersetzt in ähnlicher Weise ein beliebiges anderes Zeichen.

ProDOS hat zwei Joker-Symbole. Das erste ist das Zeichen =. Sie kön-

nen = anstelle einer beliebigen Zeichenfolge in einem Namen verwenden. Wenn Sie zum Beispiel den Dateinamen BA=D eingeben, wird der von Ihnen benutzte Befehl auf alle Dateien angewendet, die mit BA anfangen und mit D aufhören.

Das ? ist das zweite Joker-Zeichen. ? bewirkt dasselbe wie =, außer daß ProDOS bei jeder zu bearbeitenden Datei, die es findet, nachfragt, ob die entsprechende Operation auch wirklich ausgeführt werden soll. Wenn Sie nicht wollen, daß der Befehl auf diese Datei angewendet wird, und zur nächsten Datei übergehen wollen, tippen Sie N ein.

Möchten Sie zum Beispiel alle Dateien, die mit BA anfangen und mit D aufhören, in dem Inhaltsverzeichnis UNSEREDATEI des Datenträgers PRODOS löschen, geben Sie einfach /PRODOS/UNSEREDATEI/BA=D ein. Damit werden automatisch alle Dateien in diesem Inhaltsverzeichnis gelöscht, die diesen Bedingungen genügen. Das würde bedeuten, daß die Dateien BAND, BACK.BORD und BAULAND alle gelöscht würden, wenn sie in diesem Verzeichnis enthalten wären. Weil das sehr gefährlich sein kann, gibt es bei ProDOS die Option, das ? zu verwenden. Wenn Sie nur die Dateien BAND und BAULAND in dem Beispiel löschen wollen, sollten Sie das ? anstelle des Zeichens = benutzen, so daß der Pfadname sich /PRODOS/UNSEREDATEI/BA?D liest. ProDOS würde sich dann den Namen jeder Datei bestätigen lassen, bevor es diese löscht. Sie können die Datei BACK.BORD übergehen, indem Sie den Buchstaben N eintippen, wenn sie an die Reihe kommt.

Diese Joker gibt es nur in der IIe-Version von ProDOS und nur bei den Datei-Dienstprogrammen und dem Umwandlungsprogramm. Sie können nicht vom BASIC-Prompt aus oder innerhalb eines BASIC- Programms benutzt werden.

ProDOS LADEN

Wenn Sie ProDOS benutzen wollen, müssen Sie zuerst das Betriebssystem „booten“ oder laden. Das ist sehr einfach. Nehmen Sie Ihre ProDOS User's Disk, und schieben Sie sie in Laufwerk 1 Ihres Computers. Schließen Sie die Laufwerktür, und schalten Sie den Apple II ein. Sie werden ein surrendes Geräusch hören, das Licht am Laufwerk wird kurz aufleuchten, und das Hauptmenü wird auf dem Bildschirm Ihres Monitors erscheinen.

DAS HAUPTMENÜ

Ein Menü in einem Computerprogramm ist einem Menü in einem Restaurant sehr ähnlich. Der Kellner in einem Restaurant gibt Ihnen eine Menü-

karte und wartet darauf, daß Sie von den angebotenen Sachen auswählen, was Sie wünschen. Das Menü, das ProDOS Ihnen anbietet, erscheint auf dem Bildschirm Ihres Monitors. Sie sagen ProDOS, was Sie haben möchten, indem Sie Ihre Wahl über die Tastatur eingeben. Das Menü auf der ProDOS User's Disk bietet, wie in Abb. 2.1 zu sehen ist, jeweils ein einzelnes Zeichen und – hinter einem Gedankenstrich – eine Beschreibung des vorgestellten Auswahlpunkts an. Die erste Auswahlmöglichkeit des ProDOS-Menüs sieht zum Beispiel so aus:

```
? – TUTOR: PRODOS EXPLANATION
```

Das bedeutet „Tutor: Erklärung von ProDOS". Wenn Sie den Tutor benutzen wollen, drücken Sie einfach die Taste mit dem ?.

ProDOS bietet eine Reihe von Menüs, mit denen Sie auf die Dienstprogramme zugreifen können. Wenn Sie zum Beispiel die Datei-Dienstprogramme (Filer) im Hauptmenü auswählen, lädt ProDOS diese in den Speicher und bietet Ihnen ein zweites Menü an. Das geschieht, damit Sie es leicht haben, die Vielzahl der zur Verfügung stehenden Funktionen

```
***************************************
*                                     *
*          PRODOS USER'S DISK         *
*                                     *
* COPYRIGHT APPLE COMPUTER, INC. 1983 *
*                                     *
***************************************

YOUR OPTIONS ARE:

     ? - TUTOR: PRODOS EXPLANATION
     F - PRODOS FILER (UTILITIES)
     C - DOS <-> PRODOS CONVERSION
     S - DISPLAY SLOT ASSIGNMENTS
     T - DISPLAY/SET TIME
     B - APPLESOFT BASIC

PLEASE SELECT ONE OF THE ABOVE
```

Abb. 2.1: Das Hauptmenü

auszuführen. Sie werden sehen, daß dieser Ansatz es viel einfacher macht, ProDOS zu lernen und anzuwenden.

DER TUTOR

Der Tutor ist eine Option, die in Ihrem ProDOS-Menü auf jeder Stufe erscheint. Er ist ein Bestandteil, der heutzutage in der Software sehr populär ist. Manchmal wird er Help (Hilfe) oder Explain (Erklärung) anstelle von Tutor genannt, aber die Grundidee bleibt die gleiche. Er dient als schnell erreichbare Auskunftstafel, auf die Sie während der Arbeit mit den ProDOS- Dienstprogrammen zugreifen können.

In allen Fällen kommen Sie durch Drücken der ?-Taste in den Tutor. Der Tutor löscht den Bildschirm, gibt Ihnen eine kurze Beschreibung der Optionen in dem entsprechenden Menü und ein paar Tips, wie es weitergeht. Oben auf dem Bildschirm sehen Sie die Anzeige

```
TUTOR:
```

darauf folgt der Titel des Bildschirms. Wenn Sie die ?-Taste zum Beispiel im Menü der Datei-Dienstprogramme drücken, lautet der Titel

```
FILER MENU OPTIONS
```

Innerhalb des Tutors stehen Ihnen nur zwei Befehle zur Verfügung. Enthält der Tutor hier noch weitere Informationen, gibt er eine Meldung aus wie

```
PRESS <RET> TO CONTINUE: <ESC> TO EXIT
```

Wenn Sie diese Meldung auf dem Bildschirm sehen, können Sie mit der Return-Taste den Bildschirm löschen und sich die nächste Tafel mit Erklärungen zeigen lassen. Durch Drücken der Escape- Taste kehren Sie ins Hauptmenü zurück.

Haben Sie die letzte Informationstafel erreicht, die der Tutor zu diesem Thema anbietet, dann sehen Sie die Meldung

```
PRESS <ESC> TO EXIT
```

und Sie kommen nach Drücken der Escape-Taste ins Hauptmenü zurück. Keine andere Taste hat hier eine Wirkung.

Der Tutor ist für Sie besonders wertvoll, wenn Sie den Umgang mit den ProDOS-Dienstprogrammen lernen wollen. Sie können sehr oft ein Problem einfach lösen, indem Sie sich an den Tutor wenden. Wie Sie wissen, stehen die Erklärungen, die Sie auf einer Bildschirmseite des Tutors

sehen, in direktem Zusammenhang mit den Optionen, die Sie zur Verfügung haben, wenn Sie ihn aufrufen. Wenn Sie bei einem der ersten Menüs sind, erhalten Sie mehr allgemeine Informationen über ProDOS. Befinden Sie sich im Menü der Datei-Dienstprogramme, dann sehen Sie detaillierte Angaben über die Befehle, die Ihnen dort zur Verfügung stehen.

Sie sollten den Tutor benutzen, um sich schnell in ProDOS einzuarbeiten. Beginnen Sie Ihre Arbeit mit einer leeren Diskette. Probieren Sie Ihre Optionen aus. Falls Sie Probleme haben oder nicht genau wissen, was Sie tun sollen, fragen Sie den Tutor. Wenn Sie während des Lernens das System benutzen, geht es besser und schneller.

DIE DATEI-DIENSTPROGRAMME (FILER)

Wenn Sie durch Drücken von F zu den Datei-Dienstprogrammen gelangen, gibt ProDOS das Menü in Abb. 2.2 aus.

Der Tutor

Nach Drücken der ?-Taste erscheint der Tutor auf dem Bildschirm und gibt eine kurze Beschreibung Ihrer Optionen in diesem Menü. Wenn Sie den Tutor von hier aus aufrufen, bringt er nur Angaben zu den Datei-Dienstprogrammen. Ansonsten verhält er sich genau wie im Kapitel über den Tutor beschrieben.

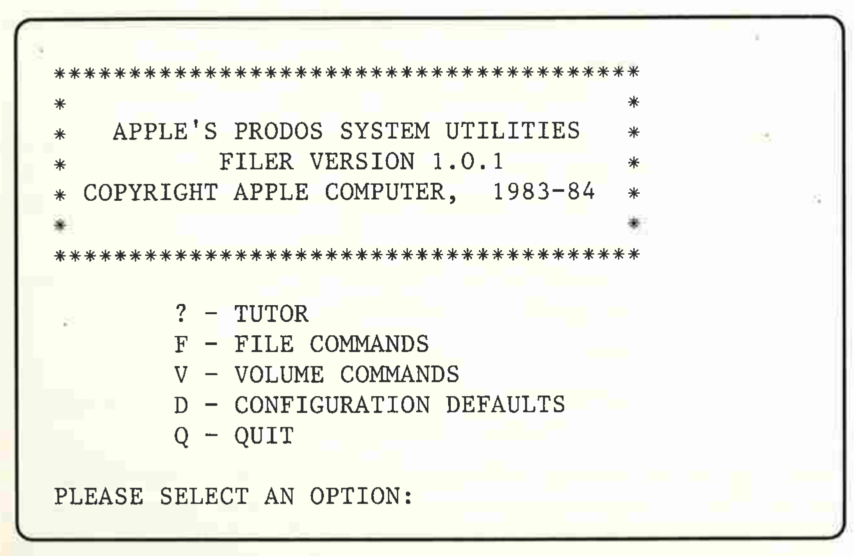


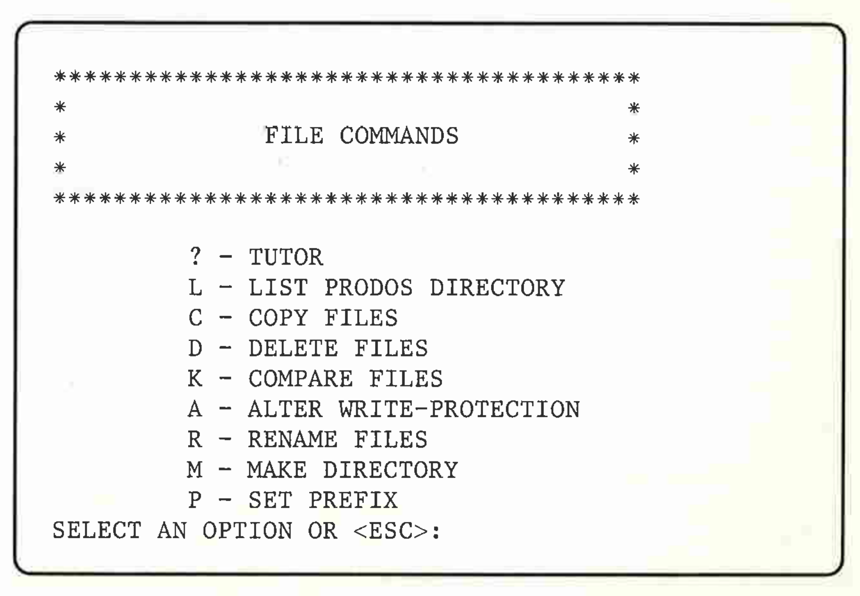
Abb. 2.2: Das Menü der Datei-Dienstprogramme

Dateibefehle

Wenn Sie F für FILE COMMANDS (Befehle zum Behandeln von Dateien) drücken, wird Ihnen ein Menü mit allen Optionen zur Behandlung einzelner Dateien vorgestellt. Abb. 2.3 zeigt dieses Menü. Zuerst kommt natürlich der Tutor. Wenn Sie hier die ?-Taste drücken, gibt Ihnen der Tutor ein paar Hinweise zu Pfadnamen, Dateinamen, Joker-Zeichen und Präfixen. Diese werden in diesem Buch in Kapitel 4 beschrieben.

Der nächste Befehl ist L für LIST PRODOS DIRECTORY (ProDOS-Verzeichnis auflisten). Dieser Befehl fragt Sie nach dem Pfadnamen des Verzeichnisses, das Sie aufgelistet haben möchten. Wenn Sie /PRODOS eintippen und die Return-Taste drücken, werden Sie etwas Ähnliches wie in Abb. 2.4 sehen. Oben auf dem Bildschirm steht das Wort DIRECTORY (Verzeichnis), dahinter der Name des Verzeichnisses, das Sie sich anschauen. Darunter sind die Namen der Dateien in diesem Verzeichnis mit einigen Angaben zu jeder einzelnen Datei aufgelistet.

Wenn vor dem Dateinamen ein Stern steht, ist die Datei schreibgeschützt. Eine schreibgeschützte Datei kann nicht gelöscht oder geändert werden. (Wie man eine Datei mit einem Schreibschutz versieht, werden wir ein paar Seiten später besprechen.)



```
****************************************
*                                      *
*             FILE COMMANDS            *
*                                      *
****************************************

         ? - TUTOR
         L - LIST PRODOS DIRECTORY
         C - COPY FILES
         D - DELETE FILES
         K - COMPARE FILES
         A - ALTER WRITE-PROTECTION
         R - RENAME FILES
         M - MAKE DIRECTORY
         P - SET PREFIX
SELECT AN OPTION OR <ESC>:
```

Abb. 2.3: Das Menü der Dateibefehle

```
DIRECTORY: /PRODOS

 NAME                TYP  BLOCKS  MODIFIED
*PRODOS              SYS      31    1-JAN-84
*BASIC.SYSTEM        SYS      21   15-NOV-83
*CONVERT             SYS      42    1-NOV-83
*FILER               SYS      51  <NO DATE>
 STARTUP             BAS       7   14-DEC-83
*BYE                 BAS       1   16-NOV-83
 TEST                BAS       1  <NO DATE>
*ABLE                BIN       4   21-NOV-83
 WINDOW              BAS       1   21-FEB-84

 BLOCKS FREE:   121       USED:  159


--PRESS <RET> TO BEGIN: <ESC> TO EXIT--
```

Abb. 2.4: Beispielausgabe des LIST PRODOS DIRECTORY-Befehls

In der Spalte hinter dem Dateinamen finden Sie einen Code aus drei Zeichen, der den Dateityp angibt. Er teilt Ihnen mit, ob Sie sich eine Programm-, eine Daten- oder eine Systemdatei anschauen.

Die dritte Spalte sagt Ihnen, wieviel Blöcke die Datei auf der Diskette belegt. Jeder Block ist aus 512 Bytes zusammengesetzt. Wenn Sie wissen möchten, wie groß eine Datei in Bytes ist, multiplizieren Sie einfach die Anzahl der Blöcke mit 512.

In der letzten Spalte steht das Datum, an dem die Datei zuletzt geändert worden ist. Wurde bei der letzten Änderung kein Datum angegeben, dann steht in dem Feld

<NO DATE>

Unten auf dem Bildschirm zeigt ProDOS Ihnen die Anzahl der Blöcke auf der Diskette, die noch frei sind, und die Anzahl der bereits belegten Blöcke.

Die Return-Taste bringt Sie an den Anfang des Auflistungsbefehls zurück, so daß Sie sich ein weiteres Verzeichnis anschauen können. Mit

der Escape-Taste kommen Sie ins Menü der Datei-Dienstprogramme zurück.

Mit dem COPY FILES-Befehl können Sie eine Datei aus einem beliebigen Verzeichnis in irgendein anderes Verzeichnis einer Diskette im gleichen oder in einem anderen Laufwerk kopieren. Dazu brauchen Sie den Pfadnamen der Datei, die Sie kopieren wollen, und den Pfadnamen des Verzeichnisses, in das Sie die Datei kopieren wollen. ProDOS fragt Sie nach diesen Pfadnamen und fordert Sie dann auf, die entsprechenden Disketten einzulegen und die Return-Taste zu drücken. Wenn die Kopie fertig ist, erscheint die Meldung

```
COPY COMPLETE
```

auf dem Bildschirm, und Sie kommen an den Anfang des COPY FILES-Befehls zurück. Falls Sie noch eine Datei kopieren möchten, können Sie die oben beschriebenen Schritte wiederholen. Sind Sie mit dem Kopieren fertig, bringt Sie die Escape-Taste wieder ins Menü der Datei-Dienstprogramme zurück.

Beim DELETE FILES-Befehl (Befehl zum Löschen von Dateien) werden Sie aufgefordert, den Pfadnamen der zu löschenden Datei anzugeben. Wenn Sie die Return-Taste drücken, schaut ProDOS in dem von Ihnen angegebenen Verzeichnis nach und löscht die Datei sofort, wenn es sie findet. Das ist eine sehr gefährliche Option – Sie sollten stets den Pfadnamen noch einmal überprüfen, bevor Sie die Return-Taste drükken. Die Datei-Dienstprogramme lassen keine doppelte Überprüfung der zu löschenden Datei zu. Wenn die Datei einmal weg ist, können Sie sie nicht mehr zurückbekommen. Der DELETE FILES-Befehl löscht keine Inhaltsverzeichnis- oder Unterverzeichnis-Dateien, es sei denn, das Verzeichnis ist leer (das heißt, es enthält keine weiteren Dateien mehr). Auf diese Weise verhindert ProDOS, daß eine Datei gelöscht wird, die Ihr einziger Zugang zu einer Menge anderer Dateien ist. Wollen Sie eine Verzeichnisdatei löschen, so müssen Sie vorher jede einzelne Datei in diesem Verzeichnis löschen.

Der COMPARE FILES-Befehl (Befehl zum Vergleichen von Dateien) überprüft, ob zwei Dateien identisch sind. Sie geben ProDOS die Pfadnamen der Dateien an, die Sie vergleichen wollen. Dieser Befehl ist nützlich, wenn Sie überprüfen wollen, ob Sie Ihre Sicherungsdatei (backup file) auf den neuesten Stand gebracht haben, nachdem Sie Ihre Arbeitsdatei geändert haben. Wenn beide Dateien identisch sind, meldet das Programm

```
COMPARE COMPLETE
```

Sind die Dateien nicht identisch, sehen Sie die Meldung

```
FILES DO NOT MATCH
```

Dieser Befehl zeigt Ihnen nicht, wo sich die beiden Dateien unterscheiden; den Unterschied müssen Sie selbst herausfinden.

Mit dem ALTER WRITE-PROTECTION-Befehl können Sie ungeschützte Dateien mit einem Schreibschutz versehen oder bei geschützten Dateien den Schreibschutz wieder entfernen. Eine schreibgeschützte Datei kann weder geändert noch gelöscht werden. Sie sollten sich angewöhnen, von dieser Möglichkeit Gebrauch zu machen, um versehentliches Löschen wichtiger Dateien zu verhindern.

Der Befehl fragt Sie zuerst nach den Pfadnamen der Dateien, die Sie ändern wollen. Wenn Sie diesen eingegeben und die Return-Taste gedrückt haben, stellt das Programm die Frage

```
LOCK FILES?(Y/N)
```

Wollen Sie die angegebenen Dateien schützen, drücken Sie Y für „Yes“. Dann wird das Verzeichnis auf den neuesten Stand gebracht, und Sie können die Datei nicht mehr ändern, löschen oder umbenennen, bis Sie ihren Status wieder geändert haben. Wenn der Vorgang beendet ist, sehen Sie die Meldung

```
LOCK COMPLETE
```

Wollen Sie den Schreibschutz entfernen, müssen Sie die Frage mit N für „No“ beantworten. Jetzt meldet das Programm

```
UNLOCK COMPLETE
```

wenn es fertig ist. Sie dürfen bei diesem Befehl Joker-Zeichen einsetzen, so daß Sie den Schreibschutz bei mehreren Dateien gleichzeitig ändern können.

Mit dem RENAME FILES-Befehl können Sie den Namen einer Datei ändern, ohne mit deren Inhalt in Berührung zu kommen. Der Befehl fragt Sie zuerst nach dem derzeitigen Pfadnamen und dann nach dem neuen Pfadnamen, die beide mit dem Dateinamen enden. Der neue Pfadname endet natürlich mit dem neuen Dateinamen. Wenn die Namensänderung fertig ist, erscheint die Meldung

```
RENAME COMPLETE
```

und Sie befinden sich wieder am Anfang des RENAME FILES-Befehls. Zum Verlassen müssen Sie wie bei allen anderen Befehlen dieses Menüs die Return-Taste drücken.

Mit dem MAKE DIRECTORY-Befehl können Sie ein neues Verzeichnis anlegen. Sie werden nach einem Pfadnamen gefragt; er muß mit dem Namen des Verzeichnisses enden, das Sie anlegen wollen. Nachdem Sie den neuen Pfadnamen eingegeben und die Return-Taste gedrückt haben, legt das Programm dieses Verzeichnis an und meldet

```
MAKE DIRECTORY COMPLETE
```

Der letzte Befehl im Menü der Dateibefehle ist SET PREFIX. Mit diesem Befehl können Sie ein Systempräfix setzen, um sich beim Eingeben eines Pfadnamens unnötige Tipparbeit zu ersparen. Tippen Sie einfach das Präfix ein, das Sie benutzen wollen, und drücken Sie die Return-Taste. Das Programm meldet

```
SET PREFIX COMPLETE
```

wenn das Präfix gesetzt ist. Danach wird das neue Präfix automatisch vor jeden Pfadnamen gesetzt, den Sie eingeben. Mit der Escape-Taste kommen Sie ins Menü der Dateibefehle zurück.

Seien Sie vorsichtig beim Gebrauch des SET PREFIX-Befehls. Wenn Sie aus irgendeinem Grund zwei Disketten mit einem ähnlichen Namen im Laufwerk haben und versehentlich das falsche Präfix setzen, laufen Sie Gefahr, einen folgenschweren Fehler zu machen. Beim Löschen von Dateien kann es Ihnen zum Beispiel passieren, daß Sie die falsche Datei entfernen, weil das Präfix auf die falsche Diskette zeigt. Der gleiche Fehler kann bei gleichlautenden Dateinamen in verschiedenen Verzeichnissen derselben Diskette passieren. Wenn Sie das Präfix auf ein spezielles Unterverzeichnis gesetzt haben, können Sie mit dem LIST PRODOS DIRECTORY-Befehl den eingegebenen Unterverzeichnis-Namen noch einmal überprüfen, bevor irgend ein Schaden angerichtet werden kann.

Datenträgerbefehle (Volume Commands)

Wenn Sie im Menü der Datei-Dienstprogramme V für VOLUME COMMANDS drücken, sehen Sie ein Menü mit allen Optionen zur Behandlung ganzer Disketten. Dieses Menü ist in Abb. 2.5 dargestellt. Die erste Option ist natürlich der Tutor. Hier gibt der Tutor einige Hinweise zu Steckplätzen (slots), Laufwerken (drives) und Datenträgernamen. Diese Begriffe sind am Anfang dieses Kapitels beschrieben worden.

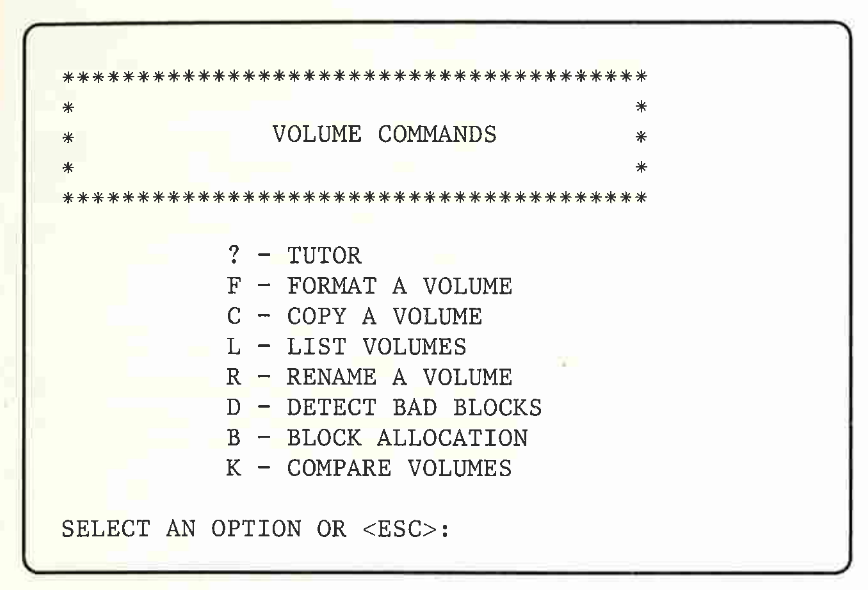


Abb. 2.5: Das Menü der Datenträgerbefehle

Der FORMAT A VOLUME-Befehl bereitet eine Diskette für den Gebrauch vor. Bei diesem Befehl wird eine Diskette in Blöcke aufgeteilt, so daß ProDOS auf ihr Daten speichern, finden und lesen kann. Zuerst werden Sie nach Steckplatz- und Laufwerknummer der Diskette gefragt, die Sie formatieren wollen. Nachdem Sie das eingegeben haben, fragt ProDOS Sie nach dem neuen Datenträgernamen. Abb. 2.6 zeigt eine Beispielausgabe für den FORMAT A VOLUME-Befehl. Wenn Sie der Diskette beim Formatieren keinen Namen geben möchten, drücken Sie einfach die Return-Taste, und das Programm wird ihr den Namen /BLANK und eine zweistellige Zahl zuweisen. Formatieren nimmt etwas Zeit in Anspruch; während dieser Zeit sehen Sie die Meldung

```
FORMATTING
```

Wenn der Formatierungsvorgang beendet ist, erscheint

```
FORMAT COMPLETE
```

und Sie kommen an den Anfang des Programms zurück.

Mit dem COPY A VOLUME-Befehl können Sie einen Datenträger auf einen anderen kopieren. Das geht aber nur bei Datenträgern, die genau gleich sind. Es ist also nicht möglich, mit diesem Befehl den Inhalt einer

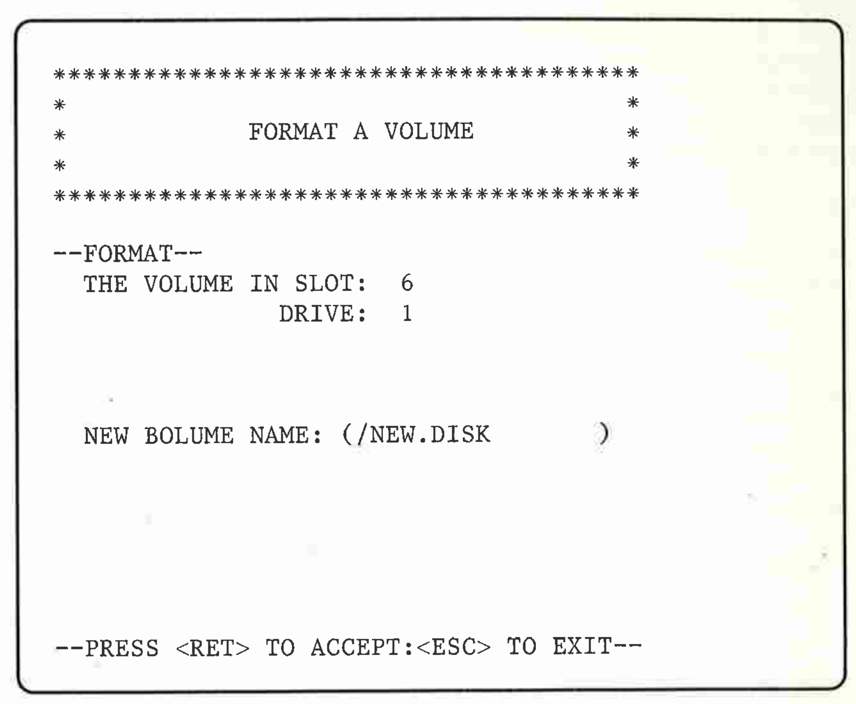


Abb. 2.6: Der FORMAT A VOLUME-Befehl

Festplatte auf eine Diskette zu kopieren oder umgekehrt, weil sie nicht das gleiche Format haben. Das geht nur mit dem COPY FILES-Befehl im Menü der Dateibefehle. Sie sollten bei der Anwendung des COPY A VOLUME-Befehls vorsichtig sein – der vorherige Inhalt der Zieldiskette geht nämlich während des Kopiervorgangs verloren.

Beim COPY A VOLUME-Befehl werden Sie zuerst nach den Nummern von Steckplatz (slot) und Laufwerk (drive) des Datenträgers, den Sie kopieren wollen, und des Datenträgers, auf den Sie kopieren wollen, gefragt. Danach werden Sie aufgefordert, die Disketten einzulegen und die Return-Taste zu drücken. Dann fragt ProDOS Sie nach dem Namen des neuen Datenträgers. Sie können den voreingestellten Namen /PRODOS akzeptieren oder ihn in einen beliebigen anderen Namen umändern. Sobald die Kopie fertig ist, meldet ProDOS

    COPY COMPLETE

und kehrt an den Anfang des COPY A VOLUME-Befehls zurück.

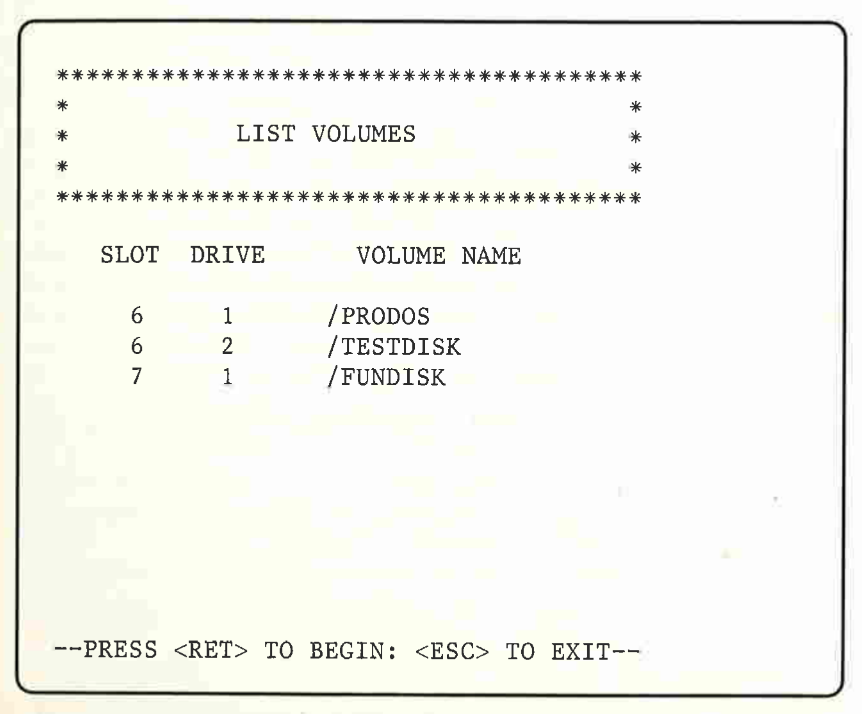
Wenn Sie nur ein Laufwerk haben, sind Steckplatz- und Laufwerknummer des Datenträgers, den Sie kopieren wollen, und des Datenträgers, auf den Sie kopieren wollen, identisch. Das Problem bei nur einem Laufwerk besteht darin, daß Sie die Disketten mehrmals austauschen müssen, bevor die Kopie fertiggestellt ist. ProDOS sagt Ihnen zuerst

INSERT SOURCE DISK AND PRESS <RET>

Legen Sie jetzt die Originaldiskette, das ist die Diskette, die Sie kopieren wollen, ins Laufwerk, und drücken Sie die Return- Taste. Dann sagt ProDOS

INSERT DESTINATION DISK AND PRESS <RET>

Legen Sie nun die Zieldiskette ins Laufwerk; das ist die Diskette, auf die Sie die Originaldiskette kopieren wollen. Wenn Sie jetzt die Return-Taste betätigen, fragt ProDOS, ob es die Daten, die sich noch auf der Zieldiskette befinden, überschreiben soll. Wenn Sie N für „No" eingeben, bricht


```
****************************************
*                                      *
*            LIST VOLUMES              *
*                                      *
****************************************

   SLOT  DRIVE      VOLUME NAME

     6      1      /PRODOS
     6      2      /TESTDISK
     7      1      /FUNDISK

--PRESS <RET> TO BEGIN: <ESC> TO EXIT--
```

Abb. 2.7: Beispielausgabe beim LIST VOLUMES-Befehl

ProDOS den Kopiervorgang ab. Antworten Sie mit Y für „Yes", fährt ProDOS mit dem Kopieren fort und fordert Sie mehrmals in der oben beschriebenen Art auf, die Disketten zu wechseln. Ist die Kopie fertig, können Sie mit der Escape-Taste ins Menü der Datenträgerbefehle zurückgehen.

Der LIST VOLUMES-Befehl gibt eine Liste der Datenträger in den angeschlossenen Laufwerken aus. Abb. 2.7 zeigt ein Beispiel einer Bildschirmausgabe bei diesem Befehl. Wie Sie sehen, sind in der Liste auch die Nummern von Steckplatz und Laufwerk eines jeden Datenträgers angegeben. Sie können die Disketten in den Laufwerken austauschen und die Return-Taste drücken, um sich eine neue Liste ausgeben zu lassen.

Der RENAME A VOLUME-Befehl fragt Sie zuerst nach Steckplatz und Laufwerk des Datenträgers, den Sie umbenennen wollen. Anschließend fragt er Sie nach dem neuen Datenträgernamen, während er Ihnen den alten Namen zeigt, wie Abb. 2.8 illustriert. Wenn Sie sich entschließen, den Namen doch nicht zu ändern, drücken Sie die Return- oder Escape-

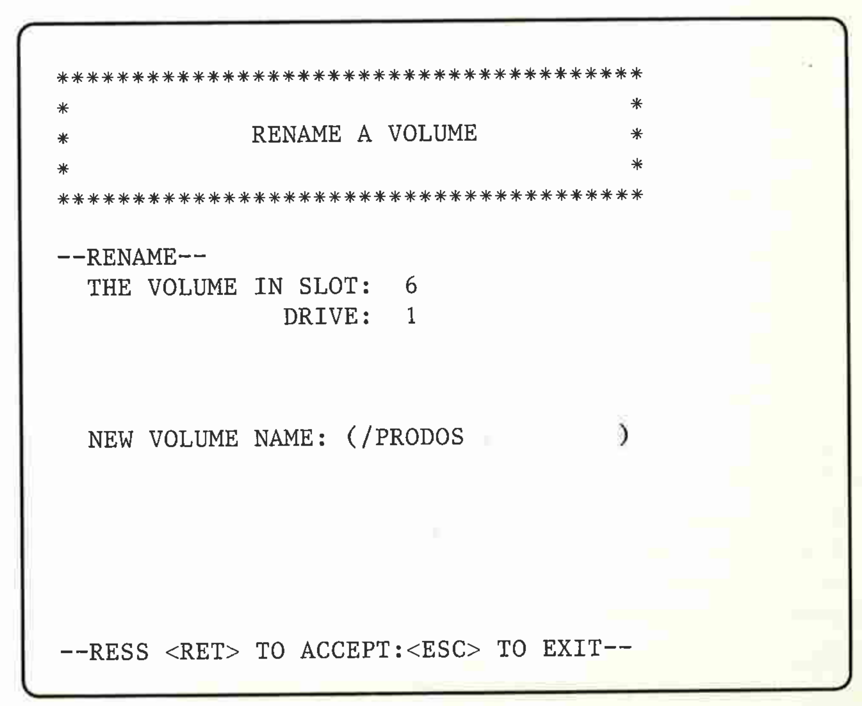


Abb. 2.8: Der RENAME A VOLUME-Befehl

Taste. Ansonsten geben Sie den neuen Namen ein (der dann den alten Namen ersetzt), und drücken Sie die Return-Taste. ProDOS wird die Meldung

```
RENAME COMPLETE
```

ausgeben und an den Anfang des Umbenennungsbefehls zurückkehren, sobald es den Namen geändert hat.

Der DETECT BAD BLOCKS-Befehl überprüft, ob es auf einem Datenträger beschädigte oder unleserliche Blöcke gibt. Bei diesem Befehl fragt ProDOS Sie nach den Nummern von Steckplatz (slot) und Laufwerk (drive) des Datenträgers, den Sie überprüfen wollen. Sobald Sie diese Fragen beantwortet haben, beginnt es zu lesen und den Datenträger zu überprüfen. Wenn es fertig ist, gibt es eine Meldung aus, die Ihnen die Anzahl der beschädigten Blöcke mitteilt, und kehrt an den Anfang zurück. Sind alle Blöcke auf der Diskette in Ordnung, sieht die Meldung so aus:

```
0 BAD BLOCKS
```

Wenn Sie eine Diskette mit beschädigten Blöcken finden, vergeuden Sie keine Zeit. Kopieren Sie sofort alle Dateien auf eine andere Diskette. Möglicherweise wird Ihnen das bei einigen Dateien wegen der beschädigten Blöcke nicht gelingen. In diesem Fall erscheint die Meldung

```
I/O ERROR
```

und Sie müssen diese Dateien aufgeben. Nachdem Sie alle Dateien, die zu retten waren, kopiert haben, formatieren Sie die beschädigte Diskette neu, und überprüfen Sie sie noch einmal.

Befinden sich auf der neu formatierten Diskette immer noch beschädigte Blöcke, sollte die Diskette als unzuverlässig betrachtet werden. Am besten werfen Sie die beschädigte Diskette weg. Müssen Sie sie trotzdem benutzen, seien Sie sehr vorsichtig. Speichern Sie keine Dateien auf ihr ab, die Sie sich nicht anderswoher wiederbesorgen können.

Ist der Datenträger, auf dem Sie die beschädigten Blöcke entdecken, eine ProFile oder eine andere Festplatte, dann haben Sie ein größeres Problem. Retten Sie durch Kopieren so viel wie möglich, und wenden Sie sich sofort an Ihren Händler oder den Hersteller.

Wenn Sie feststellen, daß mehrere Ihrer Disketten beschädigte Blöcke haben, sollten Sie untersuchen, woran es liegt. Ihr Laufwerk könnte der Verursacher sein. Probieren Sie eine andere Diskettenmarke aus. Falls

immer noch beschädigte Blöcke auftreten, muß Ihr Laufwerk möglicherweise repariert oder umgetauscht werden.

Der BLOCK ALLOCATION-Befehl (Blockzuteilungsbefehl) zeigt Ihnen, wieviel Blöcke belegt sind, wieviel Blöcke noch frei sind und wieviel Blöcke es überhaupt auf einem Datenträger gibt. Wenn Sie diesen Befehl eingeben, fragt Sie ProDOS nach den Nummern von Steckplatz (slot) und Laufwerk (drive) des Datenträgers, den Sie untersuchen wollen. Sobald Sie diese Daten eingegeben haben, macht ProDOS die entsprechenden Angaben und kehrt zum ersten Prompt zurück, wie Abb. 2.9 illustriert. Diese Angaben bleiben so lange auf dem Bildschirm, bis Sie entweder eine andere Steckplatznummer eingeben oder die Escape-Taste drücken, wenn Sie ins Menü der Datenträgerbefehle zurückgehen wollen. Mit diesem Befehl können Sie leicht nachprüfen, ob Sie auf einer Diskette noch genügend Platz haben, um noch etwas Neues darauf zu kopieren.

Mit dem COMPARE VOLUMES-Befehl können Sie zwei verschiedene Datenträger vergleichen und überprüfen, ob sie genau übereinstimmen. Dieser Befehl wird hauptsächlich benutzt, um festzustellen, ob die Siche-

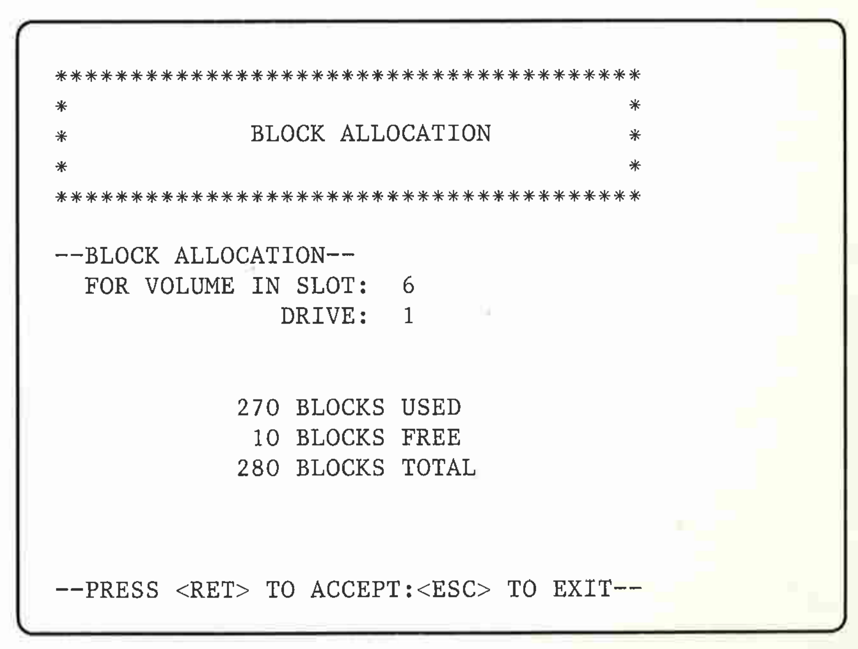


Abb. 2.9: Beispielausgabe beim BLOCK ALLOCATION-Befehl

rungskopie einer Arbeitsdiskette auf den neuesten Stand gebracht worden ist. ProDOS fragt wieder nach den Steckplatz- und Laufwerknummern der zu vergleichenden Datenträger. Sind beide Datenträger identisch, erscheint die Meldung

COMPARE COMPLETE

auf dem Bildschirm. Gibt es einen Unterschied, sehen Sie eine Meldung wie

BLOCK NUMBERS DO NOT MATCH:

wobei darunter die Nummern der nicht identischen Blöcke angegeben sind.

Konfigurations-Voreinstellungen (Configuration Defaults)

Als nächstes gibt es im Menü der Datei-Dienstprogramme die Möglichkeit, Voreinstellungen zu machen. Wenn Sie von diesem Menü aus D eingeben, erhalten Sie das in Abb. 2.10 dargestellte Menü aller Optionen zur Konfiguration der Voreinstellungen des Systems. Die erste Option ist wie gewöhnlich der Tutor. Hier erhalten Sie vom Tutor einige Hinweise über Steckplätze, Laufwerke und Datenträgernamen.

Das Programm nimmt an, daß Sie zwei Diskettenlaufwerke an eine Karte in Steckplatz 6 angeschlossen haben, daß Sie Laufwerk 1 als Quelllaufwerk und Laufwerk 2 als Ziellaufwerk haben möchten und die gesamte Ausgabe über den Monitor erfolgen soll. Das sind die sogenannten Konfigurations-Voreinstellungen; sie enthalten Angaben darüber, wie Ihr System zusammengesetzt ist. Mit der CONFIGURATION DEFAULTS-

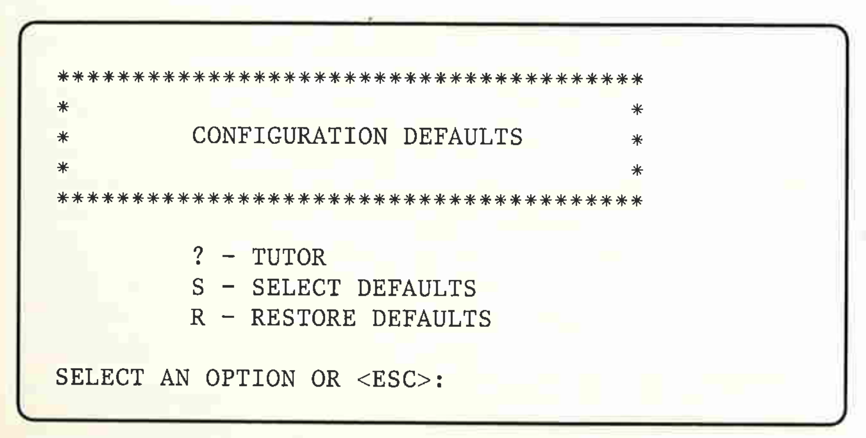


Abb. 2.10: Das CONFIGURATION DEFAULTS-Menü

Option können Sie je nach Wunsch neue Werte einstellen oder die ursprünglichen Werte wiederherstellen.

Der SELECT DEFAULTS-Befehl fragt Sie, wo Sie Quellsteckplatz und -laufwerk bzw. Zielsteckplatz und -laufwerk haben möchten und über welches Gerät die Standardausgabe erfolgen soll. Mit Hilfe dieses Befehls können Sie die Voreinstellungen so ändern, daß die gesamte Ausgabe gleichzeitig über den Drucker und den Monitor erfolgt. Wenn Sie nur ein Laufwerk haben, möchten Sie gerne, daß Quellaufwerk und -steckplatz und Ziellaufwerk und -steckplatz identisch sind. Nachdem Sie diese Voreinstellungen getroffen haben, bringt Sie das Programm an den Anfang des Befehls zurück, und Sie können den Vorgang wiederholen oder mit der Escape-Taste zum Voreinstellungsmenü zurückkehren. In Abb. 2.11 sehen Sie, wie die Bildschirmanzeige bei diesem Befehl aussieht.

Der Befehl ist sehr nützlich, wenn Sie mehr als zwei Laufwerke haben oder wenn Sie schnell eine gedruckte Liste von dem, was auf dem Bildschirm zu sehen ist, haben wollen. In einem System mit drei oder vier Laufwerken können Sie Quellaufwerk, Ziellaufwerk und das Laufwerk

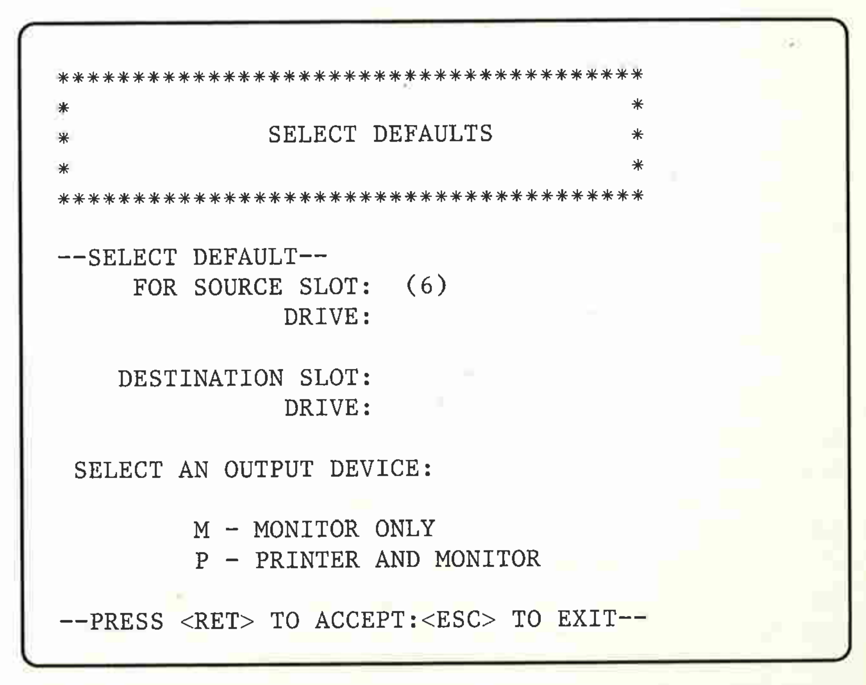


Abb. 2.11: Der SELECT DEFAULTS-Befehl

mit der ProDOS User's Disk so festlegen, daß sie alle verschieden sind, und brauchen dann auf die Aufforderungen der Datei-Dienstprogramme, die entsprechenden Disketten einzulegen, nur die Return-Taste zu drükken. Wenn Sie wollen, daß alles, was auf dem Bildschirm erscheint, als „Echo" auch über den Drucker ausgegeben wird, setzen Sie die Ausgabe auf P für Drucker und Monitor (Printer and Monitor) anstelle von M für den Monitor allein (Monitor only). Danach wird ProDOS versuchen, die gesamte Ausgabe gleichzeitig auf den Drucker und den Monitor zu übertragen. Damit können Sie viele Stunden Zeit sparen, wenn Sie sie am nötigsten brauchen; denn so können Sie in einer komplizierten Situation eine genaue gedruckte Aufzeichnung eines jeden einzelnen Schrittes anfertigen. Wenn Sie Probleme haben, können Sie viel schneller deren Ursachen finden.

Der RESTORE DEFAULTS-Befehl zeigt Ihnen die Standard-Voreinstellungen des Systems, und Sie können mit ihm alle Änderungen wieder rückgängig machen. Wenn Sie das tun wollen, müssen Sie die Return-Taste drücken; andernfalls drücken Sie die Escape-Taste zum Verlassen. Nachdem Sie eine dieser beiden Tasten gedrückt haben, kehrt das Programm ins Menü der Voreinstellungen zurück.

Verlassen (Quit)

Der letzte Befehl im Menü der Datei-Dienstprogramme ist der QUIT-Befehl. Er fragt Sie nach dem Pfadnamen des nächsten Programms und gibt einen voreingestellten Wert an. Sie können einen neuen Pfadnamen eintippen oder durch Drücken der Return- Taste die Voreinstellung übernehmen. Wenn Sie zuerst ProDOS in Ihr System geladen haben, lautet die Voreinstellung zum Verlassen des Programms BASIC.SYSTEM. Falls Sie den Namen eines anderen Programms eintippen, bleibt diese neue Voreinstellung erhalten, bis Sie sie wieder ändern oder ProDOS neu laden. Mit der Escape- Taste kommen Sie ins Hauptmenü zurück.

UMWANDLUNG VON DOS NACH ProDOS

Eine der wichtigsten Eigenschaften von ProDOS ist die Möglichkeit, unter DOS 3.3 geschriebene Programme zu benutzen. Normalerweise können Sie die gleichen Programme und die gleichen Daten unter beiden Betriebssystemen verwenden. Apple hat Ihnen ein Dienstprogramm zur Umwandlung von einem Format ins andere mitgeliefert.

Wenn Sie dieses CONVERT-Dienstprogramm durch Drücken von C im Hauptmenü starten, wird das Menü wie in Abb. 2.12. ausgegeben. Oben auf dem Bildschirm sehen Sie Angaben über den gegenwärtigen Transferstatus. In der Mitte des Schirms, zwischen den beiden Linien, stehen Ihre

```
               CONVERT Menu
Direction: DOS 3.3 S6,D2  ---> ProDOS
Date: <NO DATE>
Prefix: /PRODOS/

---------------------------------------------------

   R - Reverse Direction of Transfer
   C - Change DOS 3.3 Slot and Drive
   D - Set ProDOS Date
   P - Set ProDOS Prefix
   T - Transfer (or List) Files

---------------------------------------------------

Enter Command: ?  ? - Tutor,  Q - Quit
```

Abb. 2.12: Das CONVERT-Menü

Optionen. Das Prompt zum Eingeben Ihrer Befehle erscheint unten auf dem Schirm zusammen mit zwei weiteren Wahlmöglichkeiten, dem ? für den Tutor und dem Q zum Verlassen.

Transferangaben

Die erste Angabe für die Umwandlung ist die Richtung. Wenn Sie zuerst ins CONVERT-Menü kommen, ist die Umwandlungsrichtung von DOS 3.3 nach ProDOS gesetzt, wobei die DOS-Diskette ins Laufwerk 2 und die ProDOS-Diskette ins Laufwerk 1 gehören. ProDOS teilt Ihnen das durch die Meldung

```
DOS3.3 S6,D2 ---> ProDOS
```

mit. Die Steckplatznummer der DOS-Diskette ist 6, was durch S6 angezeigt wird, und ihre Laufwerknummer ist 2, wie durch D2 angegeben wird. Wenn Sie die Übertragungsrichtung umdrehen, ändert sich diese Meldung entsprechend.

Die zweite Zeile zeigt Ihnen das Datum, das verwendet wird, um die Dateien bei der Übertragung mit einem Zeitstempel zu versehen. Wenn Sie keine Uhr/Kalender-Karte besitzen und das ProDOS-Datum nicht gesetzt haben, steht in diesem Feld <NO DATE>. Andernfalls wird dort das Datum angezeigt, beispielsweise 20-MAR-85. Dieses Feld wird jedesmal, wenn Sie das Datum ändern, auf den neuen Stand gebracht.

Die dritte Angabe ist das Präfix, das Sie gerade benutzen. ProDOS zeigt es Ihnen durch eine Meldung wie /PRODOS/. In diesem Fall würde die Übertragung auf einen Datenträger mit Namen PRODOS erfolgen. Wenn Sie die SET PRODOS PREFIX-Option (Option zum Setzen des Präfix) benutzen, ändert sich diese Meldung, um das neue Präfix anzuzeigen.

Optionen im Umwandlungsmenü

Mit der REVERSE DIRECTION OF TRANSFER-Option sagen Sie ProDOS, daß Sie in die umgekehrte Richtung transferieren wollen. Zu Anfang ist die Übertragungsrichtung von DOS 3.3 nach ProDOS gesetzt. Wenn Sie R im CONVERT-Menü drücken, ändert sich die Anzeige in der DIRECTION-Zeile oben auf dem Bildschirm. Durch nochmaliges Drücken der R-Taste stellen Sie die ursprüngliche Richtung wieder her. Diese DIRECTION-Zeile sagt Ihnen immer, in welche Richtung ProDOS die Dateien transferiert.

Mit der CHANGE DOS 3.3 SLOT AND DRIVE-Option können Sie festlegen, in welches Laufwerk die DOS-Diskette kommt. Das Umwandlungsprogramm benutzt für Steckplätze und Laufwerke dieselbe Terminologie wie die Datei-Dienstprogramme. Wenn Sie C drücken, werden Sie nach Steckplatz und Laufwerk für die DOS 3.3-Diskette gefragt. Sie können durch Drücken der Return-Taste die Voreinstellungen auf dem Bildschirm übernehmen oder einen neuen Wert eintippen, wenn Sie das wollen. Mit Escape kommen Sie ins Menü des Umwandlungsprogramms zurück. Alle Änderungen, die Sie machen, spiegeln sich in den Richtungsangaben in der DIRECTION-Zeile oben auf dem Schirm wieder.

Mit der SET PRODOS DATE-Option können Sie jedes Datum eingeben, mit dem Sie die umgewandelten Dateien gestempelt haben möchten. Dieses Datum erscheint beim Auflisten eines ProDOS- Inhaltsverzeichnisses; es ist eine nützliche Hilfe, mit der Sie feststellen können, welche Datei die neuesten Daten enthält. Wenn Sie eine Uhr/Kalender-Karte in Ihrem Apple haben, liest ProDOS sie beim Ladevorgang automatisch ab, um das richtige Datum zu erhalten. Andernfalls müssen Sie das Datum per Hand eingeben. Das können Sie hier oder mit der DATE AND TIME-Option (die einige Seiten später besprochen wird) im Hauptmenü tun. Alle Änderungen, die Sie hier machen, spiegeln sich im DATE-Feld oben auf dem Bildschirm wieder. Wenn kein Datum eingestellt ist, verwendet ProDOS den Ausdruck <NO DATE>. Alle Dateien werden so lange mit diesem Ausdruck markiert, bis Sie ein Datum setzen. Sie können diese Option auch dazu benutzen, das bestehende Datum, das Pro-

DOS beim Laden abgelesen hat, zu ändern. Mit Escape kommen Sie wieder ins Umwandlungsmenü zurück.

Mit dem SET PRODOS PREFIX-Befehl können Sie zweierlei machen. Als erstes können Sie den Pfadnamen setzen, dem bei der Ausführung eines ProDOS-Befehls gefolgt werden soll. Als zweites können Sie Steckplatz und Laufwerk des an der Umwandlung beteiligten ProDOS-Datenträgers ändern.

Wenn Sie das ProDOS-Präfix durch einen neuen Pfadnamen ersetzen wollen, drücken Sie P, und tippen Sie den Pfadnamen ein, dem ProDOS folgen soll. Manchmal wissen Sie den Pfadnamen nicht genau. Wahrscheinlich wollen Sie die Dateien auf einen leeren ProDOS-Datenträger überspielen. In diesem Fall brauchen Sie nur die leere formatierte Diskette ins Laufwerk zu legen und dem Umwandlungsprogramm zu sagen, an welchem Steckplatz und in welchem Laufwerk sie sich befindet. ProDOS wird den Datenträgernamen auf der Diskette lesen und das Präfix entsprechend ändern, so daß es mit diesem Namen übereinstimmt. Alle Änderungen, die Sie hier machen, werden im PREFIX-Feld oben auf dem Bildschirm angezeigt.

Mit der TRANSFER OR LIST FILES-Option können Sie Dateien zwischen DOS und ProDOS transferieren. Sie können sich mit diesem Befehl auch die Dateien auf einer Diskette auflisten lassen. Wenn Sie also T vom Umwandlungsmenü aus drücken, werden Sie nach dem Namen der Datei gefragt, die Sie kopieren wollen. Falls Sie diesen schon kennen, können Sie ihn hier einfach eingeben. Erinnern Sie sich aber nicht genau an den Namen, dann drücken Sie die Return-Taste, um eine Liste aller Dateien auf der Originaldiskette zu erhalten. Einer der Dateinamen ist durch einen Leuchtstreifen hervorgehoben. Mit den Pfeiltasten können Sie den Leuchtstreifen zu jeder aufgeführten Datei bewegen. Dabei ist es nicht möglich, über den Anfang oder das Ende der Liste hinauszuwandern. Wenn Sie die Leertaste drücken, wird die beleuchtete Datei mit einem Pfeil links neben dem Dateinamen markiert. Haben Sie versehentlich eine falsche Datei markiert, können Sie den Leuchtstreifen zu dieser Datei zurückbewegen und die Markierung wieder entfernen, indem Sie die Leertaste noch einmal drücken. Markieren Sie auf diese Art alle Dateien, die Sie transferieren wollen. Wenn Sie danach die Return-Taste drücken, werden diese Dateien von der einen Diskette auf die andere übertragen. Ist der Kopiervorgang beendet, kommen Sie mit der Escape-Taste ins Umwandlungsmenü zurück.

Ist beim Übertragen einer Datei ein Fehler aufgetreten, erscheint unten auf dem Bildschirm eine Fehlermeldung, und das Wort ERROR steht

links von der Datei, die das Problem verursacht hat. Sie müssen das Problem beheben, bevor Sie einen neuen Versuch machen, die Datei zu kopieren. Mit Escape kommen Sie ins Hauptmenü zurück.

STECKPLATZZUWEISUNGEN (SLOT ASSIGNMENTS)

Wenn Sie diese Option des Hauptmenüs wählen, sagt ProDOS Ihnen den Namen der Diskette, mit der Sie das System gestartet haben. Es sagt Ihnen auch, ob Sie einen Apple II+ oder IIe benutzen (fälschlicherweise behauptet es, daß Sie einen IIe haben, wenn Sie die User's Disk auf einem IIc laufen lassen), wieviel Speicherplatz Sie zur Verfügung haben und ob Sie Applesoft im ROM haben. Dann zeigt es Ihnen eine Liste der Karten, die sich zur Zeit in den Erweiterungssteckplätzen des Computers befinden.

Im allgemeinen kann ProDOS erkennen, um welchen Kartentyp es sich bei jedem einzelnen Steckplatz handelt, und es teilt Ihnen diesen Typ mit. Wenn in einem Steckplatz keine Karte steckt, sagt es Ihnen, daß der Steckplatz leer ist. Befindet sich in einem Steckplatz eine Karte, dessen Typ ProDOS nicht kennt, erscheint im allgemeinen die Meldung USED, um zu zeigen, daß der Steckplatz belegt ist.

DATUM UND UHRZEIT (DATE AND TIME)

Mit dieser Option können Sie bei ProDOS Datum und Uhrzeit setzen, wenn Sie keine Uhr/Kalender-Karte haben. ProDOS hält nach einer solchen Karte Ausschau und liest dort automatisch Datum und Uhrzeit ab. Diese Information wird in erster Linie dazu benutzt, um Dateien beim Kopieren oder nach einer Überarbeitung einen Datumsstempel aufzudrücken.

Findet ProDOS keine Uhr/Kalender-Karte, steht im Datumsfeld <NO DATE>. Das erscheint dann auch in den Verzeichnislisten. Es gibt zwei Hauptgründe, eine Datei mit einem Datumsstempel zu versehen: Erstens möchten Sie bei verschiedenen Versionen einer Datei wissen, welche die zuletzt geänderte ist; zweitens möchten Sie aufzeichnen, wann ein Vorgang ausgeführt worden ist, zum Beispiel das Anfertigen einer Sicherungskopie einer Datei.

BASIC

Wenn Sie die BASIC-Option im Hauptmenü gewählt haben, befinden Sie sich im direkten Kontakt mit ProDOS. Das bedeutet, Sie brauchen nicht

durch die Menüstruktur zu gehen, um den ProDOS-Befehl einzugeben, den Sie brauchen. Diese Befehle und ihr Format werden in Kapitel 4 besprochen. Links auf dem Bildschirm neben dem Cursor sehen Sie das]-Zeichen. Es ist als BASIC- Prompt bekannt und bedeutet, ProDOS wartet darauf, daß Sie einen Befehl eingeben. Sie können hier nicht nur die ProDOS-Befehle benutzen, die über die Menüs nicht zur Verfügung stehen; Sie können, wenn Sie diese Option gewählt haben, auch in BASIC programmieren. Programmierer werden diese Option wählen, um ihre BASIC-Programme zu schreiben und laufen zu lassen. Weil sich mehrere spätere Kapitel in diesem Buch mit dem Programmieren unter ProDOS beschäftigen, werden wir das hier nicht behandeln.

Von hier aus können Sie auch ein Anwendungsprogramm starten. Viele Software-Pakete sind auch ohne die ProDOS User's Disk lauffähig. Sie werden über ein eigenes Menü gesteuert, nachdem das System von der Diskette geladen wurde, die das Programm enthält.

Sie können ins Hauptmenü zurückkehren, indem Sie RUN STARTUP eintippen und die Return-Taste drücken. Dabei muß sich die ProDOS User's Disk im Laufwerk befinden.

Kapitel 3

Die System-Dienstprogramm-Diskette für den Apple IIc

In diesem Kapitel erklären wir, wie die Standardoptionen verwendet werden, die das Menü der ProDOS-System-Dienstprogramme auf der Diskette für den Apple IIc zur Verfügung stellt. Hiervon gibt es eine Fassung in deutscher Sprache. Die Optionen in diesem Menü unterscheiden sich in mehrfacher Hinsicht von den Optionen im Menü auf der ProDOS User's Disk für den IIe. Die beiden Versionen werden im folgenden IIe- und IIc-Version genannt.

Vielleicht ist es schwierig zu verstehen, warum das gleiche Betriebssystem in zwei verschiedenen Versionen erscheint. ProDOS selbst ist genau gleich. Der Unterschied liegt nicht im Betriebssystem, sondern in den Dienstprogrammen, die Apple mitgeliefert hat, damit Sie die Möglichkeiten von ProDOS besser ausnutzen können.

Wenn Sie einen IIc-Computer mit einem IIe vergleichen, werden Sie feststellen, daß diese vom Aufbau her verschieden sind. Der IIc-Computer ist kleiner. In seinem Innern ist kein Platz für Erweiterungssteckplätze wie beim IIe. (In Wirklichkeit erlischt das Garantierecht, wenn Sie einen Apple IIc öffnen.) Statt dessen hat der IIc Eingänge auf seiner Rückseite (wo man Geräte anschließen kann). Um die Kompaktheit eines Apple IIc zu erreichen, mußten die Apple-Designer einiges weglassen, um Platz zu sparen und trotzdem die wichtigsten Funktionen eines Apple II-Computers zu erhalten.

Das Menü der System-Dienstprogramme für den IIc spiegelt diese Unterschiede wieder. Trotzdem bleiben viele ProDOS-Funktionen der IIe-Version erhalten. Die IIc-Version ist im allgemeinen leichter zu handhaben, bietet aber weniger Funktionen an und ist deshalb mehr für den gele-

gentlichen Benutzer geeignet. Wie in diesem Kapitel später noch erläutert wird, läuft die System-Dienstprogramm-Diskette auf keinem anderen Apple II, die ProDOS User's Disk läuft jedoch auch auf dem IIc mit nur unwesentlichen Schwierigkeiten.

AUSDRÜCKE UND DEFINITIONEN

Die folgenden Ausdrücke und Definitionen bilden eine wichtige Grundlage dafür, die Anwendung von ProDOS auf Ihrem Apple zu verstehen. Sie werden in diesem und im nächsten Kapitel oft benutzt. Deshalb wäre es eine gute Idee, etwas Zeit darauf zu verwenden, sich mit ihnen vertraut zu machen, bevor Sie weiterlesen. Manche Ausdrücke werden bei einem Apple IIc etwas anders benutzt als bei einem IIe oder II+; wir haben in diesen Fällen auf den Unterschied hingewiesen.

Datenträger (Volumes)

Von ProDOS aus gesehen ist ein Datenträger eine Diskette oder eine Festplatte wie etwa die ProFile. (Bis jetzt gibt es noch nicht die Möglichkeit, eine ProFile an einen Apple IIc anzuschließen.) Jeder Datenträger hat einen Datenträgernamen, der ihm während des in diesem Kapitel beschriebenen Formatierungsprozesses zugewiesen wird. ProDOS benutzt diesen Namen, um die Diskette, die sich im Laufwerk befindet, zu erkennen und um Daten zu suchen, wenn Sie dazu den Befehl geben. (Um Ihnen die Angelegenheit etwas zu erleichtern, können Sie in den System-Dienstprogrammen bei einem Befehl entweder ein Laufwerk oder einen Datenträgernamen angeben.)

Anschlüsse, Laufwerke und Steckplätze

Ein Apple IIc hat keine Erweiterungssteckplätze wie der IIe oder der II+. Statt dessen gibt es auf seiner Rückseite Anschlüsse, in die bestimmte Geräte eingestöpselt werden können. Beim Programmieren entsprechen diese Anschlüsse bestimmten Steckplätzen in anderen Apple II-Computern. Eine Gegenüberstellung sehen Sie in Tabelle 3.1.

ProDOS verwendet Steckplatznummern auf ähnliche Weise, wie die Post Namen von Staaten gebraucht. Wenn ProDOS etwas mit einem Laufwerk tun soll, muß es wissen, wo sich dieses Laufwerk befindet. Nach Angabe der Steckplatznummer weiß es, an welcher Stelle im Computer es zu suchen hat. Obwohl Sie keine Steckplatznummern brauchen, wenn Sie die Menü-Funktionen auf der System-Dienstprogramm-Diskette benutzen, müssen Sie sich doch über die Zusammenhänge zwischen Eingängen

Apple IIc	**Apple IIe, II+, II**
Anschluß 1	Steckplatz 1
Anschluß 2	Steckplatz 2
eingebaute 80-Zeichen-Karte	Steckplatz 3
Joystick-Anschluß	Steckplatz 4
eingebautes Laufwerk	Steckplatz 6, Laufwerk 1
externes Laufwerk	Steckplatz 6, Laufwerk 2

Tabelle 3.1: Apple IIc/IIe Anschluß-/Laufwerkvergleich

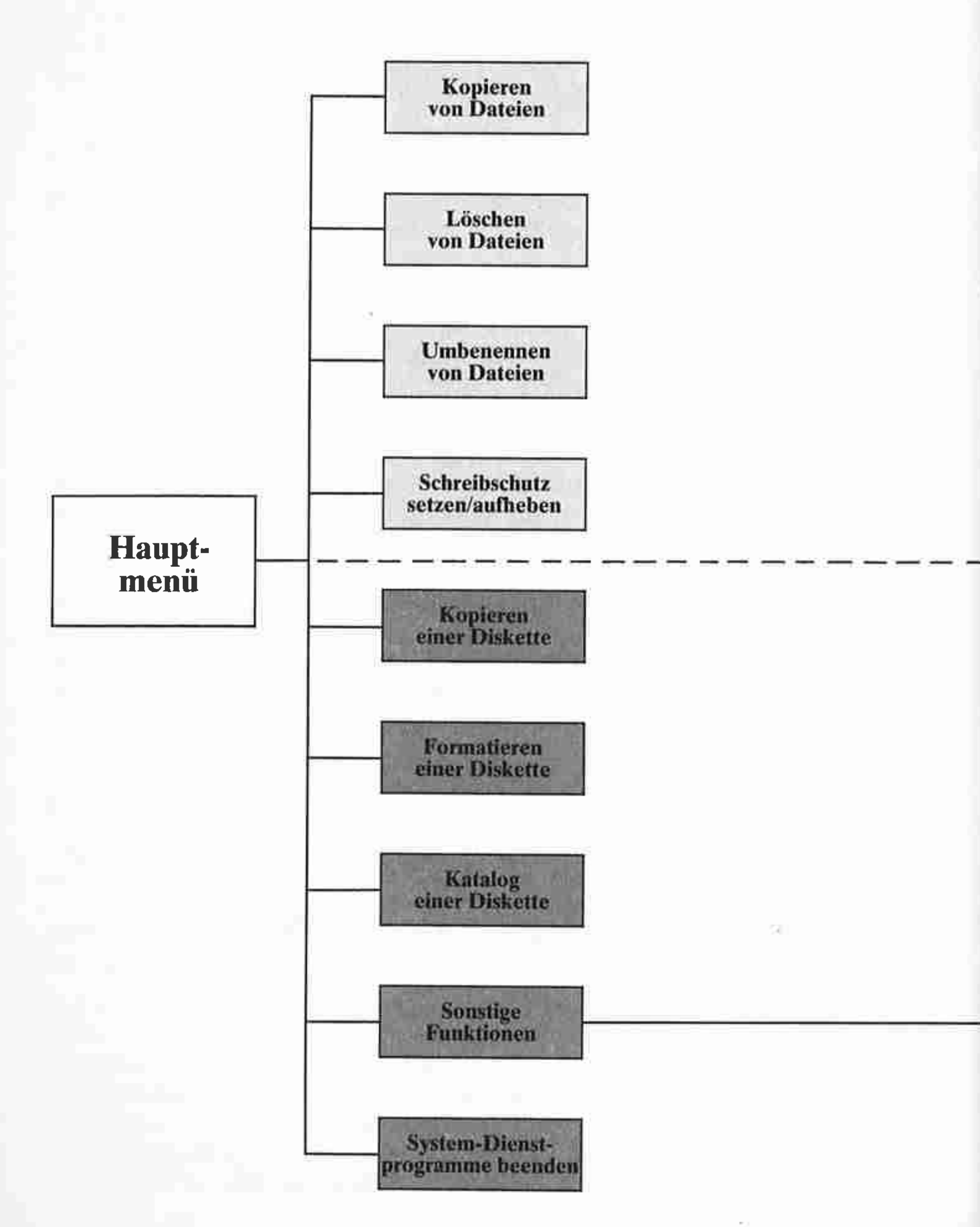
und Steckplätzen im klaren sein, wenn Sie ProDOS-Befehle vom BASIC aus aufrufen. Außerdem könnten Sie ein für die IIe-Version geschriebenes Programm nach den Nummern von Steckplatz und Laufwerk fragen.

Wie Sie in Tabelle 3.1 sehen, behandelt der IIc das interne Laufwerk als Steckplatz 6, Laufwerk 1. Ein Laufwerk, das an den Eingang für externe Laufwerke angeschlossen ist, wird als Steckplatz 6, Laufwerk 2 behandelt. Es gibt eine Ausnahme von dieser Regel: Der später in diesem Buch beschriebene PR#-Befehl kann unter diesem Arrangement nicht auf ein externes Laufwerk zugreifen. Das wäre dann wichtig, wenn das eingebaute Laufwerk funktionsuntüchtig ist, weil Sie wegen dieses Problems ProDOS nicht mehr vom eingebauten Laufwerk aus laden können. Bei einem IIe würden Sie nur die Kabelverbindungen zu Ihren Laufwerken vertauschen, wenn Sie ein Problem am Laufwerk selbst vermuten. Dann können Sie nämlich vom zweiten Laufwerk aus laden, das ja jetzt als Laufwerk 1 angeschlossen ist. Bei einem IIc geht das nicht.

Um einen völligen Stillstand zu verhindern, stellt Apple folgende Prozedur zur Verfügung:

1. Legen Sie die Diskette mit den System-Dienstprogrammen in das externe Laufwerk.
2. Drücken Sie gleichzeitig die Control- und die Reset-Taste auf Ihrem IIc.
3. Tippen Sie PR#7 ein, und drücken Sie die Return-Taste.

Diese Prozedur zwingt Ihren IIc, von der Diskette im externen Laufwerk zu laden, und Sie haben eine Möglichkeit, Ihren IIc weiter zu benutzen, obwohl das eingebaute Laufwerk nicht funktioniert. In Wirklichkeit übersetzt Ihr IIc den PR#7-Befehl (der sich in einem IIe-System auf Steckplatz 7 beziehen würde) in Steckplatz 6, Laufwerk 2.


Arbeiten an einzelnen Dateien
Kopieren von Dateien
Löschen von Dateien
Umbenennen von Dateien
Schreibschutz setzen/aufheben
Haupt-menü
Kopieren einer Diskette
Formatieren einer Diskette
Katalog einer Diskette
Sonstige Funktionen
System-Dienst-programme beenden

Die ProDOS-System-Dienstprogramme für den Apple IIc

Arbeiten mit der ganzen Diskette

- Prefix setzen
- Sub-Directory erstellen
- Konvertieren des Diskettenformats
- Überprüfen einer Diskette
- Serielle Anschlüsse konfigurieren

Pfadnamen

Pfadnamen verhalten sich auf allen Apple II-ProDOS-Systemen genau gleich und haben genau dieselbe Funktion. Pfadnamen sagen ProDOS, über welche Route es die Datei oder die Dateien, die Sie haben möchten, suchen soll. Ein Pfadname fängt immer mit einem / und dem Diskettennamen an. Der Rest des Pfadnamens besteht aus einer Folge von Verzeichnisnamen, die durch das Zeichen / getrennt sind. Der letzte Teil des Pfadnamens ist der Name der Datei, auf die Sie wirklich zugreifen wollen.

Nehmen wir zum Beispiel an, Sie möchten eine Datei mit dem Namen MEINEDATEI auf einer Diskette namens PRODOS kopieren. Diese Datei liegt innerhalb einer anderen Datei (einer Verzeichnisdatei) mit Namen UNSEREDATEI. Sie müssen einen Pfadnamen bilden, um ProDOS zu sagen, wie es sie finden kann. In diesem Fall wäre der Pfadname /PRODOS/UNSEREDATEI/MEINEDATEI. Der Pfadname kann viel länger sein, wenn MEINEDATEI innerhalb einer Reihe von Unterverzeichnissen von UNSEREDATEI liegt. Wenn MEINEDATEI im Wurzelverzeichnis (Inhaltsverzeichnis der Diskette) liegt, lautet der Pfadname einfach /PRODOS/MEINEDATEI.

Die Namen der Diskette, der Verzeichnisse und der wirklichen Datei geben ProDOS die jeweilige Richtung an, die es kennen muß, um die gewünschte Datei zu finden. ProDOS folgt diesem Pfad fast genauso, wie sich jemand an eine Beschreibung halten würde, um Ihr Haus zu finden. Wenn Ihre Angaben stimmen, findet ProDOS das Gewünschte; wenn sie falsch sind, kommt ProDOS zurück und fragt noch einmal.

Präfixe

Pfadnamen haben ihre Vorzüge, aber je länger sie sind, desto anfälliger sind sie gegenüber Fehlern. Deshalb können Sie bei ProDOS ein Präfix abspeichern, das an alle Ihre Befehle angefügt wird. Da Sie es nur einmal eintippen, werden Sie sich beim Pfadnamen nicht so oft verschreiben. Weil es automatisch vor jeden Pfadnamen gestellt wird, den Sie angeben, erspart es Ihnen außerdem eine Menge Tipparbeit. Präfixe verhalten sich wie Pfadnamen bei allen Apple II-ProDOS-Systemen genau gleich.

Angenommen, Sie arbeiten mit den Dateien, die in einer Unterverzeichnisdatei mit Namen VERZEICHNIS2 enthalten sind. Der Pfadname für diese Datei ist /DIENSTPROGRAMME/VERZEICHNIS1/VERZEICHNIS2. Wenn das Präfix auf diesen Pfadnamen gesetzt ist und Sie die Unterverzeichnisdatei VERZEICHNIS3 in VERZEICHNIS2 katalogisieren wollen, brauchen Sie zum Beispiel in der Option „Katalog einer

Diskette“ nur VERZEICHNIS3 als Antwort auf die ProDOS-Pfadname-Option einzutippen. ProDOS setzt das Präfix automatisch vor den Dateinamen, um daraus einen Pfadnamen zu machen.

Auswahl mehrerer Dateien

In der IIc-Version von ProDOS gibt es keine Joker-Zeichen wie bei der IIe-Version in den Datei- und Umwandlungs-Dienstprogrammen. Dort konnten Sie dazu benutzt werden, um eine Zeichenfolge zu spezifizieren, die mit mehr als einem Dateinamen bei einer Operation übereinstimmt. Beim IIc wurde ein anderer Ansatz gewählt: Die Dateinamen werden auf dem Bildschirm ausgegeben. Sie müssen dann die Dateien, auf die Sie eine der Menü-Funktionen anwenden wollen, einzeln markieren. Als Alternative können Sie angeben, daß die betreffende Menü-Funktion auf alle Dateien der Diskette oder des Verzeichnisses, das Sie gewählt haben, angewendet werden soll.

Wir werden dieses Vorgehen in einem folgenden Abschnitt über das Kopieren von Dateien demonstrieren.

ProDOS LADEN

Wenn Sie mit ProDOS arbeiten wollen, müssen Sie das Betriebssystem „booten“ oder laden. Das ist ganz einfach. Nehmen Sie Ihre Diskette mit den System-Dienstprogrammen, und legen Sie diese in das eingebaute Laufwerk Ihres Computers. Schließen Sie die Laufwerktür, und schalten Sie Ihren Apple IIc ein. Sie hören ein surrendes Geräusch, das Laufwerklicht an Ihrem Computer leuchtet auf, und das Hauptmenü erscheint auf dem Bildschirm Ihres Monitors.

DAS HAUPTMENÜ

Menüs haben bei einem Computer eine ähnliche Bedeutung wie in einem Restaurant. Dort gibt Ihnen der Kellner eine Menükarte und wartet darauf, daß Sie sich von den angebotenen Sachen aussuchen, was Sie gerne haben möchten. ProDOS gibt Ihnen keine Karte, sondern stellt das Menü auf dem Bildschirm Ihres Monitors dar. Sie sagen ProDOS, was Sie haben möchten, indem Sie Ihre Wahl über die Tastatur eintippen.

Das Menü der System-Dienstprogramme unterscheidet sich vom Menü der IIe-Version von ProDOS. Es enthält Funktionen, die nur beim IIc benutzt werden können, und es fehlen einige Funktionen, die in der IIe-

Version verfügbar sind. Deshalb sind die (Ihnen auf den nächsten Seiten gezeigten) Auswahlmöglichkeiten in diesem Menü verschieden. Abb. 3.1 zeigt das Hauptmenü der IIc-Version.

Die Art und Weise, wie die Menüs bei einem IIc benutzt werden, ist auch etwas anders als in der IIe-Version. In der IIc-Version können Sie Ihre Auswahl treffen, indem Sie eine der beiden Cursor-Tasten mit dem nach oben bzw. mit dem nach unten gerichteten Pfeil benutzen. Die Option in der Bildschirmzeile, die in Großbuchstaben dargestellt ist, ist gerade angewählt. Außerdem stehen die Zeichen < und > vor und hinter dieser angewählten Option. Wenn das Menü auf dem Bildschirm erscheint, ist die erste Option angewählt, wie Abb. 3.1 zeigt.

Drücken Sie die Taste mit dem nach unten gerichteten Pfeil, ist nicht mehr Option 1, Kopieren von Dateien, sondern Option 2, Löschen von Dateien, angewählt. Wenn Sie die Taste noch einmal drücken, ist Option 3 angesprochen. Wenn Sie die Möglichkeit 5 angewählt haben, sieht der Bildschirm wie in Abb. 3.2 aus. Das geht so weiter, bis Sie die letzte Option, Nummer 9, in diesem Menü erreicht haben. Wenn Sie jetzt die Taste mit dem Pfeil nach unten noch einmal drücken, ist wieder Option 1 angewählt. So können Sie sich beliebig oft im Kreis bewegen, wenn Sie

```
System-Dienstprogramme                          Haupt-Auswahl
Version 1.0
Copyright Apple Computer, Inc., 1984

-------------------------------------------------------------

   Arbeiten mit einzelnen Dateien
   1. <KOPIEREN VON DATEIEN >
   2. Löschen von Dateien
   3. Umbenennen von Dateien
   4. Schreibschutz setzen/aufheben
   Arbeiten mit der ganzen Diskette
   5. Kopieren einer Diskette
   6. Formatieren einer Diskette
   7. Katalog einer Diskette
   8. Sonstige Funktionen
   9. System-Dienstprogramme beenden
Zur Auswahl Zahl eingeben oder ↓ / ↑,
danach ↵.

-------------------------------------------------------------

Hilfe mit  - ? oder  - ?
```

Abb. 3.1: Das Menü der ProDOS-System-Dienstprogramme für den IIc

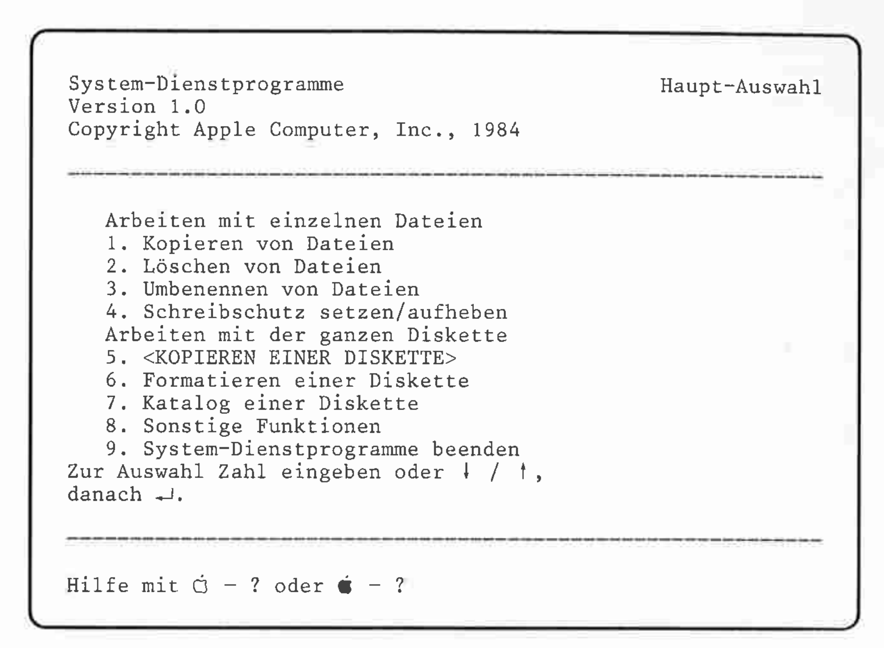


Abb. 3.2: Eine andere Option im Menü anwählen

das Ende der Liste erreicht haben. Die Taste mit dem nach oben gerichteten Pfeil hat den gleichen Effekt, außer daß die Bewegungsrichtung umgekehrt ist.

Sie können eine Option im Menü auch direkt ansteuern, indem Sie die Zahl eintippen, die vor ihr steht. Wenn Sie zum Beispiel Option 6, Formatieren einer Diskette, anwählen wollen, brauchen Sie nur die Zahl 6 einzutippen. Der Bildschirm verändert sich genauso, als ob Sie diese Option mit den Pfeiltasten angewählt hätten.

Haben Sie Ihre Wahl getroffen und wollen Sie, daß sie ausgeführt wird, dann drücken Sie die Return-Taste. Damit sagen Sie Ihrem Apple, daß Sie sich entschieden haben, und geben ihm das Signal, die Prozedur, die Sie ausgewählt haben, auszuführen.

Einige Optionen im Hauptmenü bieten Ihnen zusätzliche Auswahlmöglichkeiten in Menüform an. Diese Untermenüs folgen den gleichen Regeln wie das Hauptmenü, mit einer Ausnahme: Sie können bei diesen Untermenüs die Escape-Taste benutzen, um in das Menü zurückzukommen, in dem Sie gerade vorher gewesen sind. Diese Option wird immer durch eine kurze Meldung in der rechten oberen Ecke des Bildschirms

angezeigt, die Ihnen mitteilt, wohin Sie die Escape-Taste bringt. Wenn Sie sich zum Beispiel bei der Funktion zum Kopieren von Dateien befinden, steht in der rechten oberen Ecke des Schirms die Meldung

```
ESC: Haupt-Auswahl
```

Das bedeutet, daß Sie ins Hauptmenü zurückkommen, wenn Sie hier die Escape-Taste drücken. Mit der Escape-Taste können Sie jede Funktion verlassen, die Sie vom Hauptmenü aus erreichen können, Sie können jedoch nicht das Hauptmenü selbst damit verlassen.

HILFE

Bei der ProDOS-Version für den IIc gibt es eine Hilfe-Funktion, die Sie von überall innerhalb der System-Dienstprogramme aufrufen können. Sie nimmt den Platz des Tutors in der IIe-Version von ProDOS ein. Hierdurch besitzen Sie eine Auskunfttafel, auf die Sie während der Benutzung des Systems schnell zurückgreifen können. Sie erreichen sie, indem Sie entweder die offene oder die geschlossene Apfel-Taste gleichzeitig mit der ?-Taste drücken.

Diese Hilfe-Funktion öffnet auf dem Bildschirm ein Fenster und zeigt Ihnen eine kurze Beschreibung der Optionen in der entsprechenden Situation mit einigen Hinweisen, wie es weitergeht. Dieses Fenster enthält immer den Ausdruck „RETURN zum Fortsetzen“. Sie können weitermachen, indem Sie eine beliebige andere Taste drücken; Sie müssen nicht unbedingt die Return-Taste drücken, wenn Sie die Hilfe-Funktion verlassen wollen.

Der Inhalt des Hilfe-Fensters ändert sich entsprechend der Funktion oder des Menüs, in dem Sie sich befinden. Wenn Sie zum Beispiel im Hauptmenü sind und Kopieren von Dateien angewählt haben, enthält das Fenster eine Beschreibung dessen, was diese Funktion macht. Haben Sie die Funktion zum Umbenennen von Dateien angewählt, erscheint eine Beschreibung dieser Funktion im Fenster, wenn Sie sich helfen lassen wollen. Die Benutzung der Hilfe-Funktion ist nicht auf das Hauptmenü beschränkt. Sie können sie überall innerhalb der System-Dienstprogramme in Anspruch nehmen, um sich von ProDOS helfen zu lassen.

Sie sollten die Hilfe-Funktion benutzen, um sich schnell in ProDOS einzuarbeiten. Beginnen Sie Ihre Arbeit mit einer leeren Diskette. Probieren Sie verschiedene Optionen aus. Wenn Sie Probleme haben oder nicht wissen, was Sie machen sollen, benutzen Sie die Hilfe-Funktion. Wenn Sie während des Lernens das System benutzen, hilft es Ihnen, besser und schneller zu lernen.

ARBEITEN MIT EINZELNEN DATEIEN

Die ersten vier Optionen des Hauptmenüs bestehen aus Operationen, die einzelne Dateien betreffen:

1. Kopieren von Dateien
2. Löschen von Dateien
3. Umbenennen von Dateien
4. Schreibschutz setzen/aufheben

Wir werden sie hier einzeln diskutieren. Die gleichen Funktionen werden in der IIe-Version von ProDOS von den Dateibefehlen im Menü der Datei-Dienstprogramme (Filer) ausgeführt.

Kopieren von Dateien

Mit der Funktion zum Kopieren von Dateien können Sie eine Datei aus einem beliebigen Verzeichnis auf Ihrer Diskette in ein beliebiges anderes Verzeichnis einer Diskette im gleichen oder in einem anderen Laufwerk übertragen. Sie erfüllt die gleichen Aufgaben wie der COPY FILES-Befehl in der IIe-Version. Der einzige Unterschied liegt in der Art und Weise, in der Sie die Dateien anwählen, die Sie kopieren möchten.

Wenn Sie die diese Funktion benutzen wollen, wählen Sie zuerst Option 1 im Hauptmenü, Kopieren von Dateien, und drücken Sie die Return-Taste. Es wird Ihnen ein Menü zum Anwählen der Originaldiskette oder des Pfadnamens, von wo Sie kopieren wollen, vorgestellt. Sie können das Laufwerk mit der Originaldiskette oder einen Pfadnamen zum richtigen Verzeichnis angeben, wie das Abb. 3.3 zeigt. Wählen Sie die geeignete Option, und drücken Sie die Return-Taste. Der Pfadname muß immer auf ein Verzeichnis zeigen. Es erscheint ein ähnliches Menü, mit dem die Zieldiskette oder der Pfadname der Zieldatei angegeben wird. Treffen Sie wieder die geeignete Wahl, und drücken Sie die Return-Taste. Wenn Sie ein Unterverzeichnis angeben wollen, müssen Sie die Option für den Pfadnamen nehmen. Die Diskettenoptionen zeigen immer auf das Wurzelverzeichnis der Diskette im entsprechenden Laufwerk.

Danach werden Sie gefragt, ob Sie einige oder alle Dateien vom Quellverzeichnis ins Zielverzeichnis übertragen wollen. Die Funktion zum Kopieren von Dateien kann alle Dateien auf einer Diskette übertragen, macht das aber nicht so effektiv wie die Funktion zum Kopieren einer Diskette, die sich auch im Hauptmenü befindet. Diese Option, alle Dateien kopieren zu können, ist jedoch nützlich, wenn Sie diese in ein spezielles Unterverzeichnis übertragen wollen. Wenn Sie nur einige Dateien aus Ihrem Quellverzeichnis kopieren wollen, werden die Namen der Dateien in die-

```
System-Dienstprogramme                    Kopieren von Dateien
Version 1.0                                 ESC: Haupt-Auswahl
------------------------------------------------------------

   Wo ist die Original-Diskette?
   1. <EINGEBAUTES LAUFWERK>
   2. Externes Laufwerk
   3. ProDOS-Pfadname

Zur Auswahl Zahl eingeben oder ↓ / ↑,
danach ↵.
------------------------------------------------------------

Hilfe mit  - ? oder  - ?
```

Abb. 3.3: Ausgangsmenü der Funktion zum Kopieren von Dateien

sem Verzeichnis geladen und auf dem Bildschirm angezeigt. Folgen Sie den Anweisungen unten auf dem Bildschirm, um Dateien mit den Pfeiltasten anzusteuern. Bei den Dateien, die zum Kopieren markiert worden sind, erscheint eine Kontrollmarke links vom Dateinamen. Haben Sie alle Dateien, die Sie übertragen wollen, markiert, dann drücken Sie die Return-Taste, um anzuzeigen, daß Sie fertig sind.

Wenn Sie ein System mit zwei Laufwerken haben und dabei sind, von einem Laufwerk aufs andere zu kopieren, überträgt ProDOS einfach alle Dateien hintereinander von der Originaldiskette auf die Zieldiskette. Sind Original- und Zieldiskette oder die beiden Pfadnamen gleich, fordert ProDOS Sie jedesmal, wenn es nötig ist, auf, die Disketten zu wechseln und die Return-Taste zu drücken. Das ist auch der Fall, wenn Sie zwischen verschiedenen Verzeichnissen auf der gleichen Diskette kopieren. Wenn Sie eine Datei übertragen, deren Name schon im Zielverzeichnis existiert, fragt ProDOS Sie, ob die alte Datei gelöscht werden soll, bevor es die neue Datei an deren Stelle kopiert.

Nachdem die Kopieroperationen beendet sind, erscheint unten auf dem Bildschirm eine Meldung. Sie können die Return-Taste drücken, um weitere Dateien zu kopieren, oder die Escape-Taste, um ins Hauptmenü zurückzukehren.

Dateien löschen

Mit der Funktion zum Löschen von Dateien können Sie eine beliebige Datei aus einem beliebigen Verzeichnis einer Diskette in einem beliebi-

gen Laufwerk entfernen. Sie verhält sich genauso wie der DELETE FILES-Befehl in der IIe-Version. Wieder liegt der einzige Unterschied zwischen den beiden darin, wie Sie die Dateien anwählen, die Sie löschen wollen.

Wenn Sie diese Funktion benutzen wollen, wählen Sie Option 2 im Hauptmenü, Löschen von Dateien, und drücken die Return-Taste. Es wird Ihnen ein Menü zum Anwählen der Diskette vorgestellt, auf der gelöscht werden soll. Sie können wieder das Laufwerk festlegen oder den Pfadnamen des gewünschten Verzeichnisses angeben. Wählen Sie die geeignete Option, und drücken Sie die Return-Taste.

Sie werden dann gefragt, ob Sie einige oder alle Dateien auf dieser Diskette oder in diesem Verzeichnis löschen wollen. Wenn Sie nur einige Dateien löschen wollen, werden die Namen der Dateien auf der Diskette oder in dem Unterverzeichnis, das Sie angegeben haben, geladen und auf dem Bildschirm ausgegeben. Wählen Sie die entsprechenden Dateien mit den Pfeiltasten an; zum Löschen markierte Dateien erhalten eine Überprüfungsmarke links vom Dateinamen. Wenn Sie alle Dateien, die Sie löschen wollen, markiert haben, drücken Sie die Return-Taste. Bevor Sie eine Unterverzeichnisdatei löschen können, müssen Sie zuerst alle in ihr enthaltenen Dateien löschen.

Wenn der Löschvorgang beendet ist, erscheint eine Meldung unten auf dem Bildschirm. Drücken Sie die Return-Taste, falls Sie noch Dateien auf anderen Disketten oder in anderen Verzeichnissen löschen wollen, oder die Escape-Taste, wenn Sie zurück ins Hauptmenü wollen.

Beim Löschen von Dateien schaut ProDOS in das Verzeichnis, das Sie angeben, und entfernt eine zu löschende Datei sofort, wenn es sie findet. Das ist eine sehr gefährliche Option, und Sie sollten stets Ihre Dateinamen doppelt überprüfen, bevor Sie die Return- Taste drücken. Ist die Datei einmal weg, können Sie sie nicht mehr zurückbekommen.

Dateien umbenennen

Mit der Funktion zum Umbenennen von Dateien können Sie den Namen einer Datei ändern, ohne mit deren Inhalt in Berührung zu kommen. Obwohl die Dateien, die Sie umbenennen wollen, anders angewählt werden, verhält sich diese Funktion genau wie der RENAME FILES-Befehl in der IIe-Version.

Wenn Sie diese Funktion benutzen wollen, wählen Sie im Hauptmenü Option 3, Umbenennen von Dateien, und drücken die Return-Taste. Sie können dann entweder ein Laufwerk oder den Pfadnamen des Verzeich-

nisses angeben, in dem Sie die Änderungen machen wollen. Treffen Sie die geeignete Wahl, und drücken Sie die Return-Taste.

Danach werden Sie gefragt, ob Sie einige oder alle Dateien auf der Diskette umbenennen wollen. Wenn Sie nur einige Dateien umbenennen wollen, werden die Namen der Dateien im entsprechenden Verzeichnis geladen und auf dem Bildschirm angezeigt. Wählen Sie die gewünschten Dateien mit den Pfeiltasten an; bei diesen Dateien erscheint dann eine Überprüfungsmarke links vor dem Dateinamen. Haben Sie alle Dateien, die Sie umbenennen wollen, markiert, dann drücken Sie wieder die Return-Taste.

ProDOS fragt Sie nach einem neuen Dateinamen für jede Datei, die umbenannt werden soll. Tippen Sie den neuen Namen, den Sie der Datei geben wollen, ein, und drücken Sie die Return-Taste. Wenn Sie versuchen, einer Datei einen Namen zu geben, den es in dem Verzeichnis schon gibt, fragt ProDOS, ob Sie damit einverstanden sind, daß die alte Datei gelöscht wird. Ist das Umbenennen beendet, sehen Sie unten auf dem Bildschirm eine Meldung. Drücken Sie die Return-Taste, falls Sie noch Dateien auf anderen Disketten umbenennen wollen, oder die Escape-Taste, wenn Sie zurück ins Hauptmenü wollen.

Schreibschutz setzen/aufheben

Die Funktion zum Setzen und Aufheben des Schreibschutzes entspricht dem ALTER WRITE-PROTECTION-Befehl in der IIe-Version. Mit dieser Funktion können Sie eine Datei so markieren, daß sie nicht mehr geändert oder gelöscht werden kann, oder die Marke wieder entfernen, so daß das Löschen oder Ändern der Datei wieder möglich ist. Sie sollten sich angewöhnen, von dieser Funktion Gebrauch zu machen, um zufälliges Löschen wichtiger Dateien zu verhindern.

Wenn Sie also diese Funktion benutzen wollen, wählen Sie Option 4 im Hauptmenü, Schreibschutz setzen/aufheben, und drücken Sie die Return-Taste. Sie können entweder ein Laufwerk oder den Pfadnamen des Verzeichnisses angeben, in dem Sie den Schreibschutz ändern wollen. Wählen Sie die geeignete Option, und drücken Sie die Return-Taste.

Danach werden Sie gefragt, ob Sie einen Schreibschutz setzen oder aufheben wollen. Wählen Sie, und drücken Sie die Return-Taste. Sie haben zur Auswahl, entweder einige oder alle Dateien in dem Verzeichnis anzusprechen. Im ersten Fall werden die Namen der Dateien in dem entsprechenden Verzeichnis geladen und auf dem Bildschirm ausgegeben. Wählen Sie die zu ändernden Dateien mit den Pfeiltasten an. Bei den Dateien,

bei denen der Schreibschutz gesetzt oder aufgehoben werden soll, erscheint eine Überprüfungsmarke links vom Dateinamen. Sind Sie mit dem Auswählen fertig, drücken Sie die Return-Taste.

Nachdem alle Änderungen gemacht sind, erscheint unten auf dem Bildschirm eine Meldung. Drücken Sie die Return-Taste, falls Sie auf einer anderen Disketten oder in einem anderen Verzeichnis noch weitere Änderungen durchführen wollen, oder die Escape-Taste, wenn Sie zurück ins Hauptmenü wollen.

ARBEITEN MIT DER GANZEN DISKETTE

Die zweite Gruppe von Optionen besteht aus Operationen zum Behandeln ganzer Disketten. Wieder liefert Apple dem IIc-Benutzer vier Funktionen, die diese Aufgaben erfüllen:

5. Kopieren einer Diskette
6. Formatieren einer Diskette
7. Katalog einer Diskette
8. Sonstige Funktionen

Option 8, Sonstige Funktionen, stellt Ihnen ein zweites Menü von Funktionen zur Verfügung. Diese Funktionen werden nicht so oft gebraucht wie die anderen Optionen im Hauptmenü und erfordern etwas mehr Wissen über ProDOS und über Ihren Apple. Jeder kann sie jedoch anwenden, wenn er bereit ist, etwas Zeit und Überlegung zu investieren, um sie zu lernen. Wir werden alle diese Funktionen hier einzeln besprechen.

Kopieren einer Diskette

Die Funktion zum Kopieren einer Diskette entspricht dem COPY A VOLUME-Befehl in den Datei-Dienstprogrammen der IIe-Version. Mit dieser Funktion können Sie den Inhalt einer ganzen Diskette auf eine zweite Diskette in einer einzigen Operation kopieren.

Wenn Sie diese Funktion benutzen wollen, wählen Sie im Hauptmenü zuerst Option 5, Kopieren einer Diskette, und drücken die Return-Taste. Dann wird Ihnen ein Menü zum Auswählen des Laufwerks der Originaldiskette vorgestellt. Wählen Sie die geeignete Option, und drücken Sie die Return-Taste. Jetzt erscheint ein Menü zum Auswählen des Laufwerks der Zieldiskette. Treffen Sie die geeignete Wahl, und drücken Sie die Return-Taste. Wenn die beiden Laufwerke identisch sind, wird ProDOS Sie mehrmals auffordern, Original- und Zieldiskette auszutau-

schen. Sind sie verschieden, dann wird ProDOS den Inhalt der Originaldiskette ohne Unterbrechung auf die Zieldiskette übertragen, bis die Kopie fertig ist.

Danach erscheint unten auf dem Bildschirm eine Meldung, und Sie können die Return-Taste drücken, falls Sie noch weitere Kopien anfertigen wollen. Mit der Escape-Taste kommen Sie ins Hauptmenü zurück.

Formatieren einer Diskette

Die Funktion zum Formatieren einer Diskette bereitet eine Diskette zur Benutzung vor. Hierbei wird die Oberfläche der Diskette in Blöcke unterteilt, so daß ProDOS Daten speichern, suchen und lesen kann. Diese Funktion ist dem FORMAT A VOLUME-Befehl in der IIe-Version ähnlich, jedoch kann die IIc-Version auch DOS 3.3- und Pascal-Disketten formatieren. Sie kann sogar angeben, wie eine Diskette formatiert ist, wenn Sie den Typ nicht mehr genau kennen.

Wenn Sie diese Funktion benutzen wollen, wählen Sie Option 6, Formatieren einer Diskette, und drücken Sie die Return-Taste. Es wird Ihnen ein Menü zum Angeben des Laufwerks vorgestellt, in dem sich die Diskette befindet. Wählen Sie die geeignete Option, und drücken Sie die Return-Taste.

Sie werden dann gefragt, für welches Betriebssystem Sie die Diskette formatieren wollen. Zur Wahl stehen ProDOS, DOS 3.3 und Pascal. Wenn Sie nicht wissen, wie eine Diskette formatiert ist, können Sie Option 4 wählen, die „Ich weiß nicht"-Option. ProDOS fordert Sie dann auf, die Diskette ins Laufwerk zu legen und die Return-Taste zu drücken. Es liest die Diskette und bestimmt das Format danach, wie die Diskette bisher formatiert war. (Wenn die Diskette noch nicht oder für ein anderes Betriebssystem formatiert ist, ist ProDOS natürlich nicht in der Lage, sie zu identifizieren, und wird Ihnen das auch mitteilen.)

ProDOS fordert Sie nun auf, den Namen einzugeben, den Sie der Diskette geben wollen. Wenn Sie nur die Return-Taste drücken, wird ProDOS ihr den Namen BLANK mit einer zweistelligen Zahl dahinter zuweisen.

Ist die Diskette schon formatiert, werden Sie gefragt, ob Sie damit einverstanden sind, daß der alte Inhalt gelöscht wird. Wenn Sie nein sagen, werden Sie an den Anfang des Vorgangs zurückgeführt.

Nachdem das Formatieren beendet ist, können Sie die Return-Taste drücken, falls Sie noch weitere Disketten formatieren wollen, oder die Escape-Taste, wenn Sie zurück ins Hauptmenü wollen.

Katalog einer Diskette

Die Option 7, Katalog einer Diskette, ist dem LIST DIRECTORY-Befehl im Menü der Datei-Dienstprogramme der IIe-Version ähnlich. Diese Funktion zeigt Ihnen alle Dateien auf einer angegebenen Diskette oder in einer ProDOS-Unterverzeichnisdatei. Sie identifiziert die Diskette, indem sie ihren Namen und ihr Format angibt. Als Beispiel zeigen wir in Abb. 3.4 den Katalog der Diskette mit den System-Dienstprogrammen.

In der IIc-Version genügen drei Schritte, um sich den Katalog einer Diskette zeigen zu lassen:

1. Wählen Sie Option 7 im Hauptmenü.
2. Geben Sie ein Laufwerk oder den Pfadnamen zu dem Verzeichnis an, das Sie sich anschauen möchten.
3. Sie werden gefragt, wohin der Katalog ausgegeben werden soll. Sie können ihn sich auf dem Bildschirm oder über den Drucker ausgeben lassen. Wenn Sie die zweite Möglichkeit gewählt haben, werden Sie gefragt, an welchem Eingang Ihr Drucker angeschlossen ist. Sobald Sie diese Frage beantwortet haben, wird der Katalog Ihrer Diskette ausgedruckt werden.

In der ersten Spalte des Katalogs sind die Namen der Dateien angegeben. Wenn vor einem Namen ein Stern steht, ist diese Datei schreibgeschützt. (Eine schreibgeschützte Datei kann nicht gelöscht oder geändert werden.)

Die nächste Spalte enthält eine kurze Beschreibung, die Ihnen den Typ der Datei mitteilt. Sie erfahren hier, ob Sie sich zum Beispiel eine Programmdatei, eine Datendatei oder eine Systemdatei anschauen. Wir werden auf die einzelnen Dateitypen in Kapitel 4 eingehen. Anders als beim IIe wird beim IIc der Dateityp nicht durch einen Code aus drei Zeichen, sondern durch eine explizite Benennung angegeben. Das bedeutet nicht, daß der IIc andere Dateitypen als der IIe verwendet oder daß er zusätzliche Angaben in einem Verzeichnis speichert. Das Programm, das von der Diskette liest, übersetzt nur den Code, den es findet, in eine besser lesbare Form. Eine Datei, die beim Katalogisieren der Diskette mit den IIc-System-Dienstprogrammen als „Binary“ bezeichnet wird, wird zum Beispiel von den Datei-Dienstprogrammen des IIe als BIN bezeichnet.

Die Spalte für die Länge sagt uns, wie groß eine Datei ist, das heißt, wieviel Blöcke sie auf der Diskette belegt. Jeder Block besteht aus 512 Bytes.

```
System-Dienstprogramme                      Katalog einer Diskette
Version 1.0                                     ESC: Haupt-Auswahl
--------------------------------------------------------------

Disketten-Name: /D31.05.85             Disketten-Format: ProDOS
Dateiname                  Typ           Länge

 USER.PROFILE             Text             1
*STARTUP                  ABasic           4
*SU                       ABasic          35
*SU1.OBJ                  Binary          28
*SU2.OBJ                  Binary          10
*SU3.OBJ                  Binary          61
*SU4.OBJ                  ABasicV         19
*PRODOS                   ProDOS          31
*BASIC.SYSTEM             ProDOS          21

9 Dateien, 210 Blöcke belegt, 63 frei.
--------------------------------------------------------------

Listing komplett; drücken Sie RETURN, um weiterzumachen, ESC,
um ins Hauptmenü zurückzukehren.
```

Abb. 3.4: Beispielanzeige eines Disketten-Katalogs

Wenn Sie wissen wollen, wie groß eine Datei in Bytes ist, brauchen Sie nur die Anzahl der Blöcke mit 512 zu multiplizieren.

Unten auf dem Bildschirm gibt Ihnen ProDOS die Anzahl der aufgelisteten Dateien und die Anzahl der Blöcke an, die auf der Diskette noch frei sind.

Wenn es mehr zu zeigen gibt, als auf eine einzelne Bildschirmseite paßt, sehen Sie nach Drücken der Return-Taste die nächste Seite mit Informationen. Passen die Angaben auf eine Seite oder sind Sie auf der letzten Seite angelangt, dann kommen Sie mit der Return-Taste an den Anfang des Befehls zurück, wo Sie ein neues Verzeichnis oder ein neues Laufwerk angeben können. Ins Hauptmenü können Sie zurück, indem Sie die Escape-Taste drücken.

Sonstige Funktionen

Die ersten beiden Optionen im Menü für die sonstigen Funktionen betreffen nur ProDOS-Disketten und -Operationen. Die nächsten beiden sind auch für Disketten vorgesehen, die für DOS 3.3 formatiert sind. Die

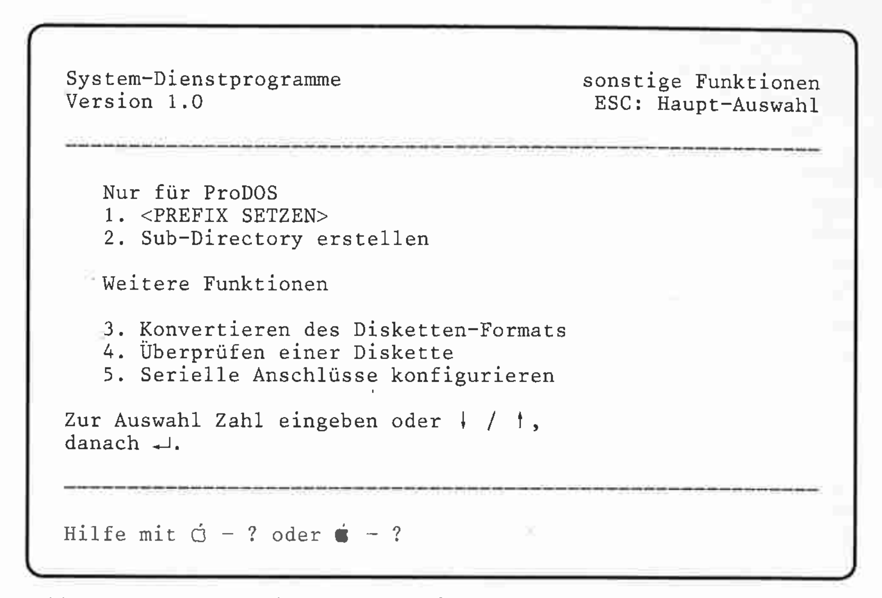


Abb. 3.5: Das Menü für die sonstigen Funktionen

letzte Option betrifft den Apple IIc direkt und handelt von den Anschlüssen der Peripheriegeräte (wie Drucker und Modem) zum Computer. Abb. 3.5 zeigt diese Optionen.

Präfix setzen

Mit der ersten Option im Menü für die sonstigen Funktionen können Sie das System-Präfix ändern, um sich unnötige Tipparbeit bei der Angabe eines Pfadnamens zu ersparen. Sobald Sie das gemacht haben, setzt ProDOS automatisch das neue Präfix an den Anfang eines jeden Pfadnamens, den Sie eingeben. Diese Option hat denselben Effekt wie der SET PREFIX-Befehl in den Datei- Dienstprogrammen der IIe-Version.

Führen Sie die folgende Schritte aus, wenn Sie bei der IIc- Version das Präfix setzen wollen:

1. Wählen Sie Option 1 im Menü für die sonstigen Funktionen.
2. Geben Sie ein Laufwerk an, oder wählen Sie die Option, einen Pfadnamen einzugeben.
3. Wenn Sie in Schritt 2 ein Laufwerk angegeben haben, liest ProDOS die Diskette in dem von Ihnen gewählten Laufwerk und setzt das Prä-

fix so, daß es auf das Wurzelverzeichnis dieser Diskette zeigt. Wenn Sie die zweite Möglichkeit gewählt haben, müssen Sie den genauen Pfadnamen eingeben, den Sie benutzen wollen. Dies ist die einzige Möglichkeit, das Präfix auf ein Unterverzeichnis zeigen zu lassen.

Seien Sie vorsichtig beim Setzen des Präfixes. Wenn Sie aus irgendwelchen Gründen zwei Disketten mit ähnlichen Namen in Ihren Laufwerken haben und das Präfix versehentlich auf den falschen gesetzt haben, laufen Sie Gefahr, einen schwerwiegenden Fehler zu machen. Das liegt daran, daß ProDOS immer den ersten passenden Namen nimmt, den es findet, was möglicherweise nicht von Ihnen beabsichtigt war. Wenn Sie zwei Disketten mit gleichem Namen in Ihren Laufwerken haben, schaut ProDOS zuerst in dem Laufwerk nach, von dem es geladen worden ist. (Das ist bei einem IIc das eingebaute Laufwerk, es sei denn, Sie haben das System von dem externen Laufwerk mit dem PR#-Befehl geladen.) Wenn Sie zum Beispiel gerade dabei sind, Dateien zu löschen, kann es Ihnen passieren, daß Sie die falsche Datei entfernen, weil das Präfix auf die falsche Diskette zeigt. Der gleiche Fehler kann bei gleichnamigen Dateien in verschiedenen Verzeichnissen auf einer Diskette auftreten. Wenn Sie in einem bestimmten Unterverzeichnis arbeiten wollen, können Sie das Präfix auf dieses zeigen lassen und sich dann mit Option 7, Katalog einer Diskette, vergewissern, daß Sie sich im richtigen Unterverzeichnis befinden, bevor Sie irgendeinen Schaden anrichten können.

Ein Unterverzeichnis (Subdirectory) erstellen

Mit der zweiten Option können Sie eine neue Unterverzeichnisdatei anlegen. Unterverzeichnisdateien sind dazu da, andere Dateien zu enthalten, eingeschlossen anderer Unterverzeichnisdateien. So können Sie mehr Dateien auf einer Diskette haben, als das Wurzelverzeichnis fassen kann. (Ein Wurzelverzeichnis kann höchstens 55 Dateien enthalten.) Dieser Befehl hat die gleiche Wirkung wie der MAKE DIRECTORY-Befehl in den Datei-Dienstprogrammen der IIe-Version.

Halten Sie sich an die folgenden Schritte, wenn Sie bei der IIc-Version ein Unterverzeichnis erstellen möchten:

1. Wählen Sie Option 2 im Menü für die sonstigen Funktionen.
2. Geben Sie ein Laufwerk an, oder wählen Sie die Option, einen Pfadnamen zu der Diskette anzugeben, auf der Sie das Unterverzeichnis anlegen wollen.
3. Wenn Sie die zweite Option gewählt haben, müssen Sie den genauen Pfadnamen, den Sie benutzen wollen, eingeben. Das ist die einzige

Möglichkeit, ein Unterverzeichnis innerhalb eines anderen Unterverzeichnisses anzulegen. Der Pfadname, den Sie eingeben, muß auf das Unterverzeichnis zeigen, in dem Sie das neue Unterverzeichnis angelegt haben wollen.

4. Danach fordert ProDOS Sie auf, den Namen des neuen Unterverzeichnisses einzugeben. Drücken Sie die Return-Taste, wenn Sie damit fertig sind.

5. ProDOS legt jetzt die Datei an. Wenn es damit fertig ist, gibt es eine entsprechende Meldung aus. Mit der Return-Taste können Sie diesen Vorgang wiederholen; mit der Escape-Taste kommen Sie ins Menü für die sonstigen Funktionen zurück.

Konvertieren des Disketten-Formats

Eine Eigenschaft von ProDOS ist besonders wichtig: Es kann unter DOS 3.3 geschriebene Dateien benutzen. Sie können normalerweise die gleichen Programme und die gleichen Daten unter beiden Betriebssystemen verwenden. Apple hat Ihnen ein Dienstprogramm mitgeliefert, das Dateien von einem Format ins andere konvertieren kann.

Dieses Programm ist viel einfacher zu benutzen als das CONVERT-Programm der IIe-Version. Es bietet auch die Möglichkeit an, DOS 3.2-Dateien nach DOS 3.3 zu konvertieren. (DOS 3.2 ist der Vorgänger von DOS 3.3 bei den Apple- Betriebssystemen.)

Zum Konvertieren des Disketten-Formats müssen Sie Option 3 aus dem Menü für die sonstigen Funktionen wählen. Dann geben Sie in dem Menü in Abb. 3.6 den Typ der Umwandlung an, die Sie machen wollen.

Sie werden aufgefordert, entweder das eingebaute oder das externe Laufwerk für Ihre Originaldiskette zu wählen. Entscheiden Sie sich für eins, und drücken Sie die Return-Taste. Dann werden Sie nach dem Laufwerk für die Zieldiskette gefragt. Geben Sie dieses an, und drücken Sie die Return-Taste.

Wenn Sie beim Kopieren sowohl das eingebaute als auch das externe Laufwerk benutzen, fordert ProDOS Sie auf, Original- und Zieldiskette in die entsprechenden Laufwerke zu legen. Wollen Sie eine Umwandlung mit nur einem Laufwerk machen, werden Sie aufgefordert, die Originaldiskette einzulegen. Drücken Sie die Return-Taste, wenn Sie das getan haben. ProDOS liest und lädt dann die Dateinamen von der Originaldiskette. Wenn Sie nur ein Laufwerk benutzen, werden Sie jetzt aufgefordert, die Zieldiskette ins Laufwerk zu legen.

```
System-Dienstprogramme    Konvertieren des Disketten-Formats
Version 1.0                           ESC: Sonstige Funktionen

------------------------------------------------------------

   Welche Konvertierung?
   1. <DOS 3.2 → DOS 3.3>
   2. DOS 3.3 → ProDOS
   3. ProDOS → DOS 3.3

Zur Auswahl Zahl eingeben oder ↓ / ↑,
danach ↵.

------------------------------------------------------------
```

Abb. 3.6: Anfangsmenü zum Konvertieren einer Diskette

Danach fragt Sie ProDOS nach neuen Diskettennamen. Geben Sie ihn ein, und drücken Sie die Return-Taste. Wenn die Diskette schon formatiert ist, fragt ProDOS, ob Sie damit einverstanden sind, daß der alte Disketteninhalt gelöscht wird. Wählen Sie, und drücken Sie die Return-Taste. Wenn Sie Nein gewählt haben, werden Sie an den Anfang des Ablaufs zurückgeführt. Wenn Sie Ja gewählt haben, beginnt ProDOS die Diskette zu formatieren.

Danach werden die Dateien kopiert. Wenn Sie nur ein Laufwerk benutzen, werden Sie mehrfach aufgefordert, die Original- und die Zieldiskette im Laufwerk auszutauschen. ProDOS gibt eine Meldung aus, wenn der Kopiervorgang beendet ist. Sie können die Return-Taste drücken, um noch eine weitere Diskette zu konvertieren, oder die Escape-Taste, um ins Menü für die sonstigen Funktionen zurückzukehren.

Überprüfen einer Diskette

Mit Option 4, Überprüfen einer Diskette, können Sie feststellen, ob es auf einer Diskette beschädigte oder unlesbare Sektoren gibt. ProDOS liest dazu jeden einzelnen Block auf der Diskette, um zu prüfen, ob es Probleme gibt. Diese Option verhält sich genauso wie der DETECT BAD BLOCKS-Befehl in den Datei-Dienstprogrammen der IIe-Version.

Wenn beschädigte Blöcke auftauchen, dürfen Sie keine Zeit verlieren. Kopieren Sie sofort alle Dateien der Diskette auf eine andere Diskette.

Das wird Ihnen wegen der beschädigten Blöcke möglicherweise nicht bei jeder Datei gelingen; in dem Fall müssen Sie diese Datei aufgeben. Wenn Sie alle zu rettenden Dateien kopiert haben, formatieren Sie die beschädigte Diskette neu, und überprüfen Sie sie noch einmal.

Gibt es auf der neuformatierten Diskette immer noch beschädigte Blöcke, sollte sie als unzuverlässig betrachtet werden. Am besten werfen Sie die beschädigte Diskette weg. Müssen Sie sie trotzdem benutzen, dann seien Sie sehr vorsichtig. Speichern Sie auf ihr keine Dateien, die Sie sich nicht anderswoher wiederbesorgen können.

Wenn es sich bei dem Datenträger, auf dem Sie die beschädigten Blöcke entdecken, um eine ProFile oder eine andere Festplatte handelt, haben Sie ein größeres Problem. Kopieren Sie so viele Dateien wie möglich, und wenden Sie sich sofort an Ihren Händler oder den Hersteller.

Haben Sie bei mehreren Disketten beschädigte Blöcke entdeckt, dann sollten Sie die Ursache des Problems untersuchen. Es kann am Diskettenlaufwerk liegen. Probieren Sie mit einer anderen Diskettenmarke aus, ob wieder beschädigte Blöcke auftauchen. Wenn das zutrifft, muß Ihr Diskettengerät möglicherweise repariert oder ausgetauscht werden.

Wenn Sie bei der IIc-Version eine Diskette überprüfen wollen, führen Sie folgende Schritte aus:

1. Wählen Sie Option 4 im Menü für die sonstigen Funktionen.
2. Geben Sie im Menü, das jetzt erscheint, das Laufwerk an, das die zu überprüfende Diskette enthält.
3. ProDOS liest jeden Block auf dieser Diskette durch und teilt Ihnen das Ergebnis dieses Tests mit. Wenn Fehler gemeldet werden, sollten Sie den oben erklärten Schritten zur Behandlung einer unzuverlässigen Diskette folgen.
4. Sie können mit der Return-Taste nach Schritt 2 oder mit der Escape-Taste ins Menü für die sonstigen Funktionen zurückkehren.

Serielle Anschlüsse konfigurieren

In der IIe-Version von ProDOS gibt es keine Funktion, die mit Option 5, Serielle Anschlüsse konfigurieren, vergleichbar wäre. Mit dieser Option können Sie die Anschlüsse für den Drucker oder ein Modem so einstellen, daß Sie auch andere Geräte als den Apple Imagewriter-Drucker oder das 300-Baud-Modem von Apple anschließen können. Das hat nichts mit ProDOS oder mit Dateien oder Disketten zu tun. Es betrifft nur den

Computer und die Peripheriegeräte, die Sie an den Computer anschließen wollen. Die IIc-Version bietet diese Option an, weil es beim IIc keine Erweiterungssteckplätze gibt. Ein Apple IIc-Computer hat nur diese beiden seriellen Anschlüsse für Erweiterungen, und es gibt nur eine beschränkte Anzahl von Möglichkeiten, diese Anschlüsse zu konfigurieren. Um das Konfigurieren zu vereinfachen, hat Apple die Option „Serielle Anschlüsse konfigurieren" auf der Diskette mit den System-Dienstprogrammen mitgeliefert.

Ihr Apple IIc ist automatisch so konfiguriert, daß er mit einem Apple Imagewriter-Drucker an Anschluß 1 und einem 300-Baud-Modem von Apple an Anschluß 2 kommunizieren kann. Bevor irgend ein anderes Gerät über die seriellen Anschlüsse benutzt werden kann, muß ihm eine PIN (Produkt-Identifikations-Nummer) zugewiesen werden. Diese Zahl teilt dem Apple die Charakteristik des Gerätes mit, so daß er in der Lage ist, mit ihm zusammenzuarbeiten. Jede Ziffer einer PIN gibt eine besondere Charakteristik des Gerätes an. Eine vollständige Beschreibung aller dieser Charakteristiken wäre sehr technisch und würde den Rahmen diesen Buches überschreiten. Die Angaben, die man braucht, um die PIN zu bestimmen, findet man im allgemeinen in der Gebrauchsanweisung des anzuschließenden Gerätes, normalerweise in einer leicht verständlichen Tabelle. Wenn dies nicht so ist, müssen Sie sich an den Händler oder Hersteller des Gerätes wenden, wenn Sie erfahren wollen, wie der Anschluß richtig konfiguriert wird. Es folgt eine kurze Beschreibung, was die Ziffern der PIN bedeuten:

- Modus (Drucker oder Modem)
- Daten-Bits und Stop-Bits
- Baud-Rate (Geschwindigkeit der Datenübertragung)
- Parität (wird zur Überprüfung von Fehlern benutzt)
- Video-Echo (bestimmt, ob die zum Anschluß übertragenen Daten auch auf dem Bildschirm ausgegeben werden sollen)
- Zeilenvorschub (Vorrücken zur nächsten Zeile nach Betätigung der Return-Taste)
- Zeilenlänge (Anzahl der auzudruckenden Zeichen pro Zeile)

Apple hat eine Geräteliste beigefügt, in der diese Angaben, die dem IIe die notwendigen Informationen (die PIN-Zahlen) liefern, für folgende vier Geräte zu finden sind: für den Apple Imagewriter, den Akustikkoppler 300 Bd, das Apple Modem 1200 Bd und den Apple Farbplotter.

Wenn Sie diese Liste auf andere Geräte erweitern wollen (so daß Sie die Anschlüsse leicht für diese Geräte richtig konfigurieren können), können Sie entweder die Angaben, die als „Kein Gerät definiert" in der Geräteliste stehen, ersetzen oder die von Apple mitgelieferten Geräteangaben ändern. Wie das geht, werden wir etwas später erklären. Zuerst behandeln wir das Konfigurieren eines Gerätes, das schon in der Liste enthalten ist.

Nehmen wir an, Sie möchten Anschluß 1 für den Farbplotter von Apple konfigurieren. Wählen Sie zuerst Option 5 im Menü für die sonstigen Funktionen. Dann wählen Sie Option 1 im Menü, das in Abb. 3.7 gezeigt wird. (Wenn Sie die Voreinstellung übernehmen wollen, brauchen Sie nur die Return-Taste zu drücken.)

Jetzt werden Sie aufgefordert, aus dem Menü in Abbildung 3.8 das Gerät auszuwählen, das Sie anschließen wollen. Wählen Sie Option 4 für den Farbplotter von Apple.

Jetzt werden Sie gefragt, ob diese Konfiguration auf der Diskette abgespeichert werden soll. Wenn Sie diese nicht auf der Diskette abspeichern, bleibt Anschluß 1 nur für diese Sitzung am Computer für den Farbplotter von Apple konfiguriert. Wenn Sie Ihr System neu laden, ist Anschluß 1 wieder für den Imagewriter eingestellt. Speichern Sie die Konfiguration

```
System-Dienstprogramme      Serielle Anschlüsse Konfigurieren
Version 1.0                      ESC: Sonstige Funktionen
-------------------------------------------------------------

   Was wollen Sie machen?
   1. <KONFIGURIEREN ANSCHLUSS 1>
      Apple Imagewriter (166/1124)

   2. Konfigurieren Anschluß 2
      Akustik-Koppler 300 Bd (232/1111)

   3. Ändern der Geräte-Liste

Zur Auswahl Zahl eingeben oder ↓ / ↑,
danach ↵.
-------------------------------------------------------------

Hilfe mit  -? oder  -?
```

Abb. 3.7: Menü für das Konfigurieren serieller Anschlüsse

```
System-Dienstprogramme                Konfigurieren Anschluß 1
Version 1.0          ESC: Serielle Anschlüsse Konfigurieren

-------------------------------------------------------------

   Wählen Sie das Gerät für Anschluß 1:

   1. <APPLE IMAGEWRITER (166/1124)>
   2. Akustik-Koppler 300 Bd (232/1111)
   3. Apple Farb-Plotter (163/1111)
   4. Kein Gerät definiert
   5. Kein Gerät definiert
   6. Kein Gerät definiert
   7. Kein Gerät definiert

   8. PIN ist bekannt
   9. PIN ist nicht bekannt

Zur Auswahl Zahl eingeben oder ↓ / ↑,
danach ↵.

-------------------------------------------------------------

Hilfe mit  -? oder  -?
```

Abb. 3.8: Wahl eines Gerätes für Anschluß 1

auf der Diskette ab, ist Anschluß 1 automatisch für den Farb-Plotter eingestellt, wenn Sie das System neu laden.

Nehmen wir an, Sie möchten Anschluß 2 für ein Gerät konfigurieren, das nicht auf der Geräteliste steht. Wählen Sie im Menü, das in Abb. 3.7 zu sehen ist, Option 2. Wenn Sie die PIN-Zahl des Gerätes, das Sie benutzen wollen, kennen, wählen Sie Option 8 im Menü von Abb. 3.8. Sie werden aufgefordert, die neue PIN einzugeben. Tun Sie es, und drücken Sie die Return- Taste. Dann werden Sie gefragt, ob die PIN stimmt – eine Gelegenheit, die eingetippte Zahl noch einmal zu überprüfen.

Wenn Sie die PIN Ihres Gerätes nicht kennen, wählen Sie Option 9. Jetzt müssen Sie eine Reihe von Fragen beantworten, die sich mit der Charakteristik des Gerätes befassen. Nachdem Sie alle Fragen beantwortet haben, zeigt Ihnen ProDOS die Liste der Merkmale, die Sie angegeben haben, und die daraus abgeleitete PIN-Zahl. Sie werden aufgefordert, diese Angaben zu bestätigen.

Jetzt werden Sie gefragt, ob Sie diese neue Konfiguration auf der Diskette

abgespeichern und damit zur voreingestellten Konfiguration für Anschluß 2 machen wollen. Sie sollten sich darüber im klaren sein, daß das Gerät beim Abspeichern der neuen Konfiguration nicht in die Geräteliste aufgenommen wird. Wenn Sie in der Lage sein möchten, die Konfiguration der Anschlüsse leicht zu ändern (und wenn Sie dieses Gerät öfter benutzen), sollten Sie die Geräteliste editieren, um das Gerät aufzunehmen.

Dazu wählen Sie Option 3 des in Abb. 3.7 gezeigten Menüs. Nachdem Sie das zu ersetzende Gerät im Menü in Abb. 3.9 ausgewählt haben, werden Sie aufgefordert, den neuen Gerätenamen einzugeben. Tippen Sie ihn ein, und drücken Sie die Return-Taste. Dann werden Sie gefragt, ob Sie die PIN-Zahl kennen. Wenn Sie mit Ja antworten, werden Sie aufgefordert, diese Zahl einzugeben. Tippen Sie sie ein, und drücken Sie die Return-Taste. Danach werden Sie gefragt, ob die Angaben richtig sind. Wählen Sie, und drücken Sie die Return-Taste.

Wenn Sie die PIN-Zahl nicht kennen, müssen Sie die Fragen zu den Merkmalen des Gerätes beantworten, so daß der Apple diese Zahl ermitteln kann. Wieder werden Sie am Ende aufgefordert, die Angaben zu bestätigen. Wenn Sie das gemacht haben, wird das Gerät in die Liste aufgenommen.

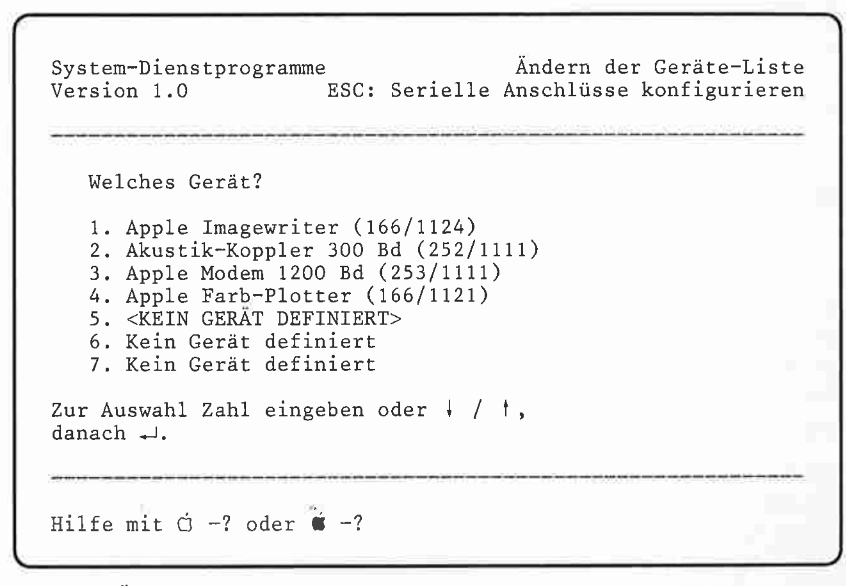


Abb. 3.9: Ändern der Geräteliste

SYSTEM-DIENSTPROGRAMME BEENDEN

Mit der letzten Option auf der Diskette können Sie die Menüstruktur der System-Dienstprogramme verlassen und Ihrem Apple über Applesoft-BASIC direkte Befehle erteilen. Wenn Sie diese Option wählen, werden Sie gefragt, ob Sie auch wirklich die System-Dienstprogramme verlassen wollen. Antworten Sie mit Ja, so kommen Sie ins BASIC, antworten Sie mit Nein, kommen Sie ins Hauptmenü zurück.

Nach der Ausführung der Option 9, System-Dienstprogramme beenden, befinden Sie sich im direkten Kontakt mit BASIC. Sie können vom BASIC aus einzelne ProDOS-Befehle eingeben; das werden wir in den nächsten Kapiteln dieses Buchs behandeln. Links auf dem Bildschirm neben dem aufleuchtenden Cursor sehen Sie ein]-Zeichen. Es ist als BASIC-Prompt bekannt und bedeutet, daß Ihr Apple auf einen Befehl wartet. Programmierer wählen diese Möglichkeit, um ihre Programme zu schreiben und laufen zu lassen. Da sich einige spätere Kapitel mit dem Programmieren unter ProDOS befassen, werden wir das hier nicht weiter diskutieren.

Von hier aus können Sie auch ein Anwendungsprogramm starten. Viele Software-Pakete laufen jedoch ohne die Diskette mit den System-Dienstprogrammen. Sie laufen über ein eigenes Menü, wenn Sie das System mit derselben Diskette laden, die das Programm enthält.

Wenn Sie zurück ins Hauptmenü wollen, tippen Sie RUN STARTUP, und drücken Sie die Return-Taste, während sich die Diskette mit den System-Dienstprogrammen im Laufwerk befindet.

DIE DISKETTE MIT DEN SYSTEM-DIENSTPROGRAMMEN UND DIE ProDOS USER'S DISK

Die für den IIc vorgesehene Diskette mit den System- Dienstprogrammen läuft nicht korrekt auf einem anderen Apple II-Computer. Das liegt an der andersartigen Hardware des IIc. Die ProDOS-Diskette für den IIc erwartet die starre und unveränderliche Konfiguration eines IIc vorzufinden – bei keinem anderen Apple findet sie alle Vorrichtungen, die sie erwartet.

Andererseits läuft die ProDOS User's Disk auch auf einem Apple IIc. Es gibt zwei kleinere Fehler, die uns bei der Benutzung dieser Version auf einem IIc aufgefallen sind. Der erste ist, daß die Option DISPLAY SLOT ASSIGNMENTS nicht erkennt, daß sie es mit einem IIc zu tun hat, und meldet, daß es sich um einen IIe handelt. Der zweite Fehler taucht beim

Umwandlungsprogramm auf. Während das Programm richtig arbeitet, ist der auf vertauschtem Hintergrund angezeigte Name der Datei, mit der Sie es gerade zu tun haben, nicht lesbar. Das hat mit den Änderungen zu tun, die Apple bei der Planung des IIc vorgenommen hat. Deshalb sind die ausgegebenen Zeichen nicht mehr zu entziffern. Dieses Problem kann nicht behoben werden.

Die IIc-Dienstprogramme sind im allgemeinen ausgefeilter im Erscheinungsbild und in der Anwendung als die Dienstprogramme auf der ProDOS User's Disk für den IIe. Die IIe-Dienstprogramme sind jedoch mächtiger und vielseitiger. Sie enthalten eine Anzahl von Möglichkeiten, die es bei den IIc-Dienstprogrammen nicht gibt. Dazu gehören die folgenden Möglichkeiten:

- Datum und Uhrzeit einstellen
- Eine Diskette umbenennen
- Alle Disketten in den Laufwerken auflisten
- Zwei Disketten vergleichen
- Konfigurations-Voreinstellungen machen
- Anzeigen, was an die einzelnen Steckplätze angeschlossen ist
- Platzvergabe und Zuteilung von Blöcken auf einer Diskette überprüfen

Wenn Sie mehr über die Dienstprogramme auf der ProDOS User's Disk wissen wollen, schlagen Sie in Kapitel 2 nach.

Kapitel 4

Verzeichnisse und ProDOS-Dateien

Ab diesem Kapitel wird ProDOS aus einer anderen Sicht betrachtet. Die vorherigen Kapitel behandelten ausführlich die Dienstprogramme, die in der IIe- und IIc-Version von ProDOS zur Verfügung stehen. Ab jetzt werden wir die Struktur von ProDOS, seine Verwendung in BASIC-Programmen und schließlich das Maschinensprache-Interface von ProDOS untersuchen. Wenn Sie nur ein gelegentlicher Benutzer sind, nur wenig oder gar nicht programmieren wollen, können Sie diese Kapitel lesen, aber die Abschnitte, die Ihnen zu technisch erscheinen, überschlagen und sich darauf konzentrieren, ein Gefühl für die Arbeitsweise von ProDOS zu bekommen. Haben Sie jedoch vor, öfter in BASIC zu programmieren, sollten Sie dieses und die nächsten Kapitel auf jeden Fall lesen, um die Möglichkeiten, die ProDOS bietet, voll auszuschöpfen. Auch als BASIC-Programmierer kann Ihnen gelegentlich ein Abschnitt in diesem Teil des Buchs zu technisch für Ihre Bedürfnisse vorkommen, wie zum Beispiel die ausführliche Behandlung eines Verzeichniskopfs und der Dateieinträge in diesem Kapitel. In diesem Fall können Sie ihn überfliegen oder auslassen und mit dem nächsten Abschnitt weitermachen. Sie können ja jederzeit nachschlagen, falls das nötig sein sollte. Der wirklich fortgeschrittene Programmierer kann umgekehrt vorgehen und den Text überfliegen, um das Material zu finden, das ihn besonders interessiert, wie zum Beispiel das Kapitel über das Maschinensprache-Interface von ProDOS oder die ausführlichen Informationen über den Inhalt einer Verzeichnisdatei und über die Dateistrukturen, die in diesem Kapitel behandelt werden.

Im vorliegenden Kapitel wird die Struktur und die Datenspeichertechnik von ProDOS untersucht. Wir behandeln ausführlich das Datenträgerkonzept (Volume), das Formatieren eines Datenträgers und wie ProDOS Dateien in Verzeichnissen und Unterverzeichnissen anlegt. Das sind

wichtige Konzepte – ein klares Verständnis ihrer Bedeutung und ihrer Arbeitsweise erleichtert Ihnen den Umgang mit ProDOS sehr.

ProDOS UND DAS FORMATIEREN VON DISKETTEN

Wie wir in Kapitel 1 erwähnt haben, können Sie die Disketten in ihren Laufwerken mit Aktenschränken in einer Registratur vergleichen, in der ProDOS der Abteilungsleiter ist. Damit ProDOS alles findet, was Sie brauchen, muß es gewisse Regeln bei der Ablage beachten. Die erste dieser Regeln ist das sogenannte Formatieren.

Die besten Aktenschränke sind sauber und ordentlich. Sie werden in einer guten Registratur keine verstreuten Hefter, falsch geordnete Aufzeichnungen oder fehlende Akten vorfinden. Die Schubfächer sind nicht überladen, und die Schränke sind so eingerichtet, daß man sie leicht benutzen kann.

So wie jemand, der in einem Büro ein neues Ordnersystem entwirft, die Einzelheiten der Organisation berücksichtigen muß, haben das auch die Designer von ProDOS getan. Bevor Sie eine Diskette unter ProDOS benutzen können, müssen Sie diese formatieren. Beim Formatieren wird der Speicherplatz auf einer Diskette so aufgeteilt, daß er vom Betriebssystem genutzt werden kann.

ProDOS stellt sich den Raum auf einer Diskette oder einer Festplatte als Datenträger vor. Es behandelt sowohl eine Diskette als auch auch eine Festplatte wie etwa die ProFile (die ungefähr 40mal so groß wie eine Diskette ist) als einen einzelnen Datenträger. Wenn Sie zwei oder mehr Laufwerke an Ihr System angeschlossen haben, betrachtet ProDOS die Disketten in diesen Laufwerken als verschiedene Datenträger. Das mag etwas ungewöhnlich scheinen, aber in einer Registratur geht es ähnlich zu. Ein kleiner Metallkasten zum Ablegen von Karteikarten ist genauso ein Datenbehälter wie ein zwei Meter hoher Schrank mit vier Schubfächern.

Damit ProDOS diese Datenträger finden kann, gibt es jedem einen Namen. Ein Datenträgername muß bestimmten Regeln genügen. Jeder Datenträgername muß mit einem Schrägstrich und einem Buchstaben anfangen. Die einzigen erlaubten Zeichen in einem Namen sind Buchstaben, Zahlen und Punkte. Es dürfen keine Zwischenräume, Umlaute oder Satzzeichen (außer dem Punkt) im Namen enthalten sein. Eine Name darf aus höchstens 15 Zeichen zuzüglich des Schrägstrichs bestehen. Hier sind einige gültige ProDOS- Datenträgernamen:

```
/DEINE.DISKETTE
/SPASS.UND.SPIEL
/DOKUMENTE
```

Nun kommen einige Namen, die ProDOS nicht akzeptiert:

```
/WILLI UND TONI'S DISKETTE
/DIE.SONNE.GEHT.AUF.
/400.SPIELE
```

Das erste Beispiel ist aus drei Gründen ungültig: Der Name hat Zwischenräume, enthält einen Apostroph und besteht aus mehr als 15 Zeichen. Der zweite Name ist einfach zu lang. Der dritte wird nicht akzeptiert, weil er mit einer Zahl anfängt; an der ersten Stelle muß ein Buchstabe stehen. Wenn Sie das vergessen und einen ungültigen Namen eintippen, gibt ProDOS eine Fehlermeldung aus.

Wie viele andere Betriebssysteme teilt DOS 3.3 eine Diskette in Spuren und Sektoren ein, die dann physikalisch auf ihr vorhanden sind. Eine Spur können Sie sich als Ring auf der Oberfläche einer Scheibe vorstellen. Ein Sektor ist ein Teil einer solchen Spur. Eine typische Diskette besitzt eine Reihe verschieden langer Spuren, die am Mittelloch anfangen und immer größer werden, je mehr sie sich dem Rand der Diskette nähern. Unter DOS 3.3 besteht jede solche Spur aus 16 Sektoren.

ProDOS arbeitet anders. ProDOS unterteilt eine Diskette in 512 Blöcke. Die Umwandlung von ProDOS-Blöcken in Diskettensektoren geht automatisch vor sich. Diese Eigenschaft macht ProDOS von allen physischen Einschränkungen bezüglich Größe und Typ einer einzelnen Diskette oder Festplatte unabhängig.

Das ist ein wichtiger Vorteil von ProDOS gegenüber DOS 3.3. Unter DOS ist es sehr schwierig, den Speicherplatz auf einem anderen Datenträger als einer normalen Diskette voll auszunutzen, weil es nur ein Format gibt. Eine 5 Megabytes große Festplatte muß als Serie von vierzig 140K-Disketten behandelt werden. Das begrenzt die Größe einer einzelnen Datei auf die Größe einer Diskette und macht es Ihnen schwerer, an die gesuchten Daten heranzukommen.

ProDOS fragt jedoch beim Gerätetreiber nach, wie groß ein einzelner Block ist. Der Gerätetreiber, der aus einer kleinen Maschinensprache-Routine besteht, die zusammen mit ProDOS in Ihr System geladen wird, übernimmt die Aufgabe, diese Angaben über eine Diskette oder Festplatte zu besorgen. Wenn eine Diskette auf 128 Bytes pro Sektor forma-

tiert ist, liest der Gerätetreiber vier Sektoren zum Ausfüllen eines Blocks; ist sie auf 256 Bytes pro Sektor formatiert, liest er nur zwei. ProDOS arbeitet unabhängig davon, wie eine Diskette organisiert ist, und kann deshalb Disketten oder Platten vieler verschiedener Größen handhaben.

Wenn ProDOS einen Datenträger formatiert, legt es die Angaben, die es zum Lesen dieses Datenträgers braucht, an einer bestimmten Stelle auf der Diskette oder Festplatte ab, damit es sie später wiederfinden kann. Diese Angaben enthalten ein Programm, mit dem ProDOS von diesem Datenträger geladen werden kann, das Stammverzeichnis und die Bitabbildung des Datenträgers. (Hinweis: Mit dem Ladeprogramm können Sie ProDOS nur dann von einem Datenträger laden, wenn sich auch die Dateien PRODOS und BASIC.SYSTEM darauf befinden.)

Das Ladeprogramm ist ein kleines Programm in Maschinensprache mit nur einer Aufgabe. Wenn Sie das System von einer Diskette in einem Laufwerk starten, wird das Ladeprogramm von der Diskette in den Speicher übertragen. Es führt dann seine Aufgabe aus, das Programm BASIC.SYSTEM von der Diskette in den Speicher zu laden.

Das Stammverzeichnis ist das Inhaltsverzeichnis des Datenträgers. Wir werden in den nächsten beiden Kapiteln genauer darauf eingehen.

Die Bitabbildung zeichnet auf, welche Blöcke auf einem Datenträger zur Zeit belegt sind. Wenn Sie eine neue Datei anlegen oder Daten zu einer bestehenden Datei hinzufügen, schaut das System in der Bitabbildung nach, wo es einen geeigneten Block zur Erweiterung der Datei finden kann. Bitabbildung heißt es, weil jeder Block auf dem Datenträger durch ein einzelnes Bit in dieser Abbildung repräsentiert wird. Wenn das Bit auf 0 gesetzt ist, ist der Block frei; ist es auf 1 gesetzt, ist der Block belegt.

DIE ORGANISATION EINES DATENTRÄGERS

Genau wie eine gute Registratur auch Angestellte hat und ein System zur Ablage der Akten benutzt, organisiert ProDOS die Daten auf einem Datenträger so, daß es leicht und schnell darauf zugreifen kann. Alle Daten sind in speziell markierten Dateien abgespeichert, damit der Inhalt leichter zu erkennen ist. Dateien können mit einem Datums- und Uhrzeitstempel versehen werden; so weiß man immer, wann sie geändert worden sind. Dateien können auch vor Änderungen geschützt werden.

ProDOS ist als hierarchisches Dateiverwaltungssystem aufgebaut. Das bedeutet, das System ist in eine Reihe von Ebenen oder Stufen aufgeteilt.

Dieses Konzept ist keine Erfindung der Computerwissenschaft; es existiert seit Tausenden von Jahren in fast jeder menschlichen Gesellschaftsordnung.

Eine Hierarchie ist mit der Struktur eines umgedrehten Baums zu vergleichen. Der Baum hat eine Wurzel. Wenn wir uns am Stamm des Baums hinunterbewegen, teilt er sich und verzweigt in eine immer größere Anzahl von Zweigen und Blättern. Die Nahrung für die Zweige und Blätter kommt über den Stamm. Eine Hierarchie gleicht einem solchen umgedrehten Baum. Jede Stufe ist an Bedeutung der Stufe über ihr untergeordnet.

ProDOS behandelt jede Diskette oder Festplatte als einen einzelnen Datenträger (die Wurzel des Baums). Jeder ProDOS- Datenträger hat ein Stammverzeichnis, das als Inhaltsverzeichnis der Diskette oder Platte dient.

Alles was Sie auf einer Diskette abspeichern, wird als Datei abgespeichert, und jede Datei hat einen Eintrag im Stammverzeichnis. Diese Dateien können Unterverzeichnisse sein, die wieder andere Dateien oder Unterverzeichnisse enthalten. ProDOS erlaubt 64 Stufen von Dateien und Unterverzeichnissen, so daß es praktisch unmöglich ist, so viele Dateien zu speichern, wie in den Verzeichnissen aufgenommen werden können.

Nehmen wir zum Beispiel an, Sie haben eine ProDOS-Festplatte mit zehn Stufen von Verzeichnissen. Die erste Stufe ist das Stammverzeichnis und auf maximal 51 Dateien begrenzt. Zehn dieser Dateien seien Unterverzeichnisdateien, und jede davon enthalte zehn weitere Unterverzeichnisse, die ihrerseits wieder zehn Unterverzeichnisse enthalten usw. bis zur Stufe zehn. Die Gesamtzahl der Dateien auf dieser Festplatte mit zehn Stufen beträgt 11 111 111 151. Die Anzahl der Dateien, die in einem einzelnen Unterverzeichnis enthalten sein können, ist nur durch den zur Verfügung stehenden Speicherplatz und die maximale Dateigröße unter ProDOS begrenzt.

DAS STAMMVERZEICHNIS

Jeder Datenträger hat ein Stammverzeichnis, das als Inhaltsverzeichnis der Diskette oder Festplatte dient. Das Stammverzeichnis ist auf 51 Einträge begrenzt. Jeder dieser Einträge kann eine Verzeichnisdatei oder eine Standarddatei sein. Wenn eine Diskette gerade formatiert worden ist, ist das Stammverzeichnis leer. Die Stammverzeichnisdatei enthält zwei Arten von Einträgen: Kopfeinträge und Dateieinträge. Diese Ein-

träge und die darin enthaltenen Information werden wir in den folgenden Abschnitten besprechen.

Der Kopfeintrag

Im Kopfeintrag eines Stammverzeichnisses sind alle wesentlichen Angaben über den Datenträger enthalten. Er enthält den Namen des Datenträgers und das Datum, an dem er angelegt worden ist. Hier stehen auch Angaben über die Version von ProDOS, mit der der Datenträger angelegt worden ist, so daß spätere ProDOS- Versionen wissen, wie sie auf ihn zugreifen können. Weiterhin sind hier die Anzahl der Verzeichniseinträge, die Anzahl der Einträge pro Block der Stammverzeichnisdatei und die Anzahl der auf diesem Datenträger zur Verfügung stehenden Blöcke aufgezeichnet. Zusätzlich enthält der Kopfeintrag noch die Blockadresse der Bitabbildung des Datenträgers.

Alle diese Angaben sind für ein effizientes Arbeiten von ProDOS notwendig. Wenn ProDOS diesen Block auf einer Diskette gelesen hat, besitzt es genügend Informationen, um jede von Ihnen gewünschte Prozedur an diesem Datenträger vorzunehmen. Die Adresse der Bitabbildung des Datenträgers, Bitabbildungszeiger (bit map pointer) genannt, macht es ProDOS leicht, den ersten freien Block zu finden. Am Namen des Datenträgers erkennt ProDOS, ob ein Pfadname dorthin zeigt. Am Anlegungsdatum können Sie erkennen, wann Sie etwas Bestimmtes gemacht haben. Hier gibt es auch ein einzelnes Byte, in dem ProDOS aufzeichnet, ob ein Datenträger schreibgeschützt ist. Alles in allem ist der Kopfeintrag mit Angaben in 39 Bytes zusammengesetzt. Diese Bytes sind in die folgenden Felder aufgeteilt.

Speichertyp und Namenslänge (1 Byte)

Das erste Byte des Kopfeintrags ist in zwei Felder von jeweils 4 Bits aufgeteilt. Die vier oberen Bits sind alle auf Eins gesetzt und zeigen an, daß es sich hier um den Schlüsselblock der Stammverzeichnisdatei handelt. Diese Angabe über den Speichertyp wird von ProDOS intern benutzt und sollte niemals geändert werden. Die vier unteren Bits enthalten die Länge des Datenträgernamens, abgespeichert als Binärzahl. Dieses Feld sollten Sie auf den neuesten Stand bringen, wenn Sie den Inhalt des Feldes für den Dateinamen unter Umgehung der ProDOS-Befehle geändert haben.

Dateinamen (15 Bytes)

Die nächsten 15 Bytes des Kopfeintrags enthalten den Namen des Datenträgers. Ein solcher Name darf aus maximal 15 Zeichen bestehen. Die in

diesem Feld abgespeicherten Daten können weniger Platz belegen (MEINEDATEI zum Beispiel belegt 10 Bytes). Das oben besprochene Feld für die Länge des Namens sagt Ihnen, wie viele dieser Bytes den wirklichen Dateinamen enthalten. Wenn Sie einen Datenträgernamen unter Umgehung der ProDOS-Befehle ändern, vergewissern Sie sich, daß Sie das Feld für die Länge des Namens angepaßt haben.

Reserviert (8 Bytes)

Die nächsten 8 Bytes werden zur Zeit nicht benutzt und sind für spätere Erweiterungen von Apple reserviert worden.

Anlegezeitpunkt (4 Bytes)

Die vier Bytes hinter dem reservierten Feld enthalten Datum und Uhrzeit des Zeitpunktes, an dem der Datenträger von ProDOS initialisiert worden ist.

Version (1 Byte)

Das nächste Byte enthält die Nummer der ProDOS-Version, mit der der Datenträger initialisiert worden ist.

Minimalversion (1 Byte)

Dieses Byte gibt die kleinste Nummer der ProDOS-Versionen an, mit denen ein Datenzugriff möglich ist.

Zugriff (1 Byte)

Hier steht, ob der Datenträger schreibgeschützt ist.

Eintragslänge (1 Byte)

Dieses Feld enthält eine Binärzahl, die die Anzahl der Bytes eines Verzeichniseintrags angibt, einschließlich des Verzeichniskopfs. Alle Verzeichniseinträge haben die gleiche Länge.

Einträge pro Block (1 Byte)

Hier wird die Anzahl der Einträge in jeden Block der Verzeichnisdatei als Binärzahl abgespeichert.

Dateizahl (2 Bytes)

Dieses Feld enthält die Anzahl der in diesem Verzeichnis eingetragenen aktiven Dateien. Eine aktive Datei ist eine Datei, die nicht den Speicher-

typ Null hat. Das bedeutet, daß alle Standard- und Verzeichnisdateien hier mitgezählt werden.

Bitabbildungszeiger (2 Bytes)

Dieses Feld enthält die Adresse des ersten Blocks der Bitabbildung des Datenträgers. Die Bitabbildung belegt immer aufeinanderfolgende Blöcke, je einen für alle angebrochenen 4096 Blöcke des Datenträgers. (Eine normale Diskette benutzt nur einen Block für die Bitabbildung. Wenn eine solche Diskette aus 4100 Blöcken bestände, würde sie zwei Blöcke für die Bitabbildung benutzen: einen für die ersten 4096 Blöcke und einen für die letzen vier Blöcke.) Der Bitabbildungszeiger verweist auf die Stelle, wo die Bitabbildung des Datenträgers auf der Diskette anfängt.

Blockanzahl (2 Bytes)

Dieses Feld enthält eine Binärzahl, die die Gesamtzahl der Blöcke auf diesem Datenträger darstellt.

Die Dateieinträge

Der zweite Eintragstyp im Stammverzeichnis heißt Dateieintrag. Für jede in dem Verzeichnis enthaltene Datei gibt es einen Dateieintrag. Die Dateieinträge enthalten Angaben, mit denen ProDOS alles finden kann, was Sie haben möchten; Sie können sie mit dem Gebäudeplan in einem Bürohaus vergleichen. Die meisten dieser Felder erscheinen auf dem Bildschirm, wenn Sie sich mit den ProDOS-Datei-Dienstprogrammen ein Verzeichnis ausgeben lassen oder wenn Sie vom BASIC-Prompt aus den CAT- oder den CATALOG-Befehl benutzen. Die Informationen, die Sie bei der Ausgabe eines ProDOS-Verzeichnisses nicht sehen, werden von ProDOS intern benutzt. Wenn Sie keine fortgeschrittenen Programme schreiben, die das Maschinensprache-Interface benutzen, kommen Sie damit nicht in Berührung. Falls Sie das nicht interessiert, können Sie die nächsten Seiten überschlagen und beim Abschnitt mit dem Titel „Unterverzeichnisdateien“ weiterlesen.

Namenslänge und Speichertyp (1 Byte)

Die ersten vier Bits des Dateieintrags enthalten die Länge des Dateinamens. Die letzten vier Bits in diesem Byte enthalten einen Wert, der den Typ der Datei angibt. Im folgenden sind die gültigen Dateityp-Codes aufgezählt (das $-Zeichen gibt an, daß der Code als Hexadezimalzahl angegeben ist):

Code	Dateityp
$1	nicht indizierte Datei (Standard seedling file)
$2	einfach indizierte Datei (Standard sapling file)
$3	zweifach indizierte Datei (Standard tree file)
$4	Unterverzeichnisdatei

Unterverzeichnisdateien haben ein ähnliches Format wie Stammverzeichnisdateien. Wir werden sie kurz besprechen. Alle anderen Dateien sind Standarddateien; ihre Struktur wird unter der entsprechenden Überschrift in diesem Kapitel erklärt.

Dateiname (15 Bytes)

Die nächsten 15 Bytes enthalten den Namen, den Sie der Datei beim Anlegen zugewiesen haben. Ein Dateiname ist unter denselben Bedingungen gültig wie ein Datenträgername. Das oben besprochene Feld für die Namenslänge muß die wirkliche Länge des Dateinamens wiedergeben. Wenn die Namenslänge 6 ist und der Dateiname aus 10 Zeichen besteht, werden immer nur die ersten sechs Zeichen verwendet. (Aus MEINEDATEI würde zum Beispiel MEINED werden.) Wenn Sie einen Dateinamen mit einem ProDOS-Befehl ändern, paßt ProDOS das Feld für die Namenslänge automatisch an.

Dateityp (1 Byte)

Dieses Byte enthält einen Code aus drei Zeichen für den Dateityp, den Sie beim Auflisten eines Verzeichnisses sehen. Eine vollständige Liste aller Dateitypen finden Sie in Anhang A.

Ein einzelnes Byte ist aus acht Bits zusammengesetzt und kann eine binäre Zahl von 0 bis 255 (einschließlich) enthalten. ProDOS verschlüsselt diese Information, um Platz auf der Diskette zu sparen. Es weiß zum Beispiel, daß die Zahl 252 (FC in hexadezimaler Schreibweise) eine Applesoft-BASIC-Datei darstellt.

Schlüsselzeiger (2 Bytes)

Das achtzehnte und das neunzehnte Byte des Dateieintrags enthalten die Blockadresse, mit der ProDOS die in diesem Eintrag beschriebene Datei lokalisiert, dargestellt als Binärzahl.

Belegte Blöcke (2 Bytes)

Die nächsten beiden Bytes teilen Ihnen mit, wieviel Blöcke die beschriebene Datei belegt. Bei einer Standarddatei entspricht das der wirklichen

Dateilänge und allem, was zusätzlich in ihr enthalten ist. Bei einer Unterverzeichnisdatei steht hier nur die Länge des Unterverzeichnisses, nicht die Länge aller in dem Unterverzeichnis enthaltenen Dateien.

EOF (Dateiende) (3 Bytes)

Diese drei Bytes stellen die Gesamtzahl der Bytes dar, die ProDOS aus der Datei lesen kann. Die Zahl ist in binärer Form gespeichert.

Im Spezialfall einer dünn besetzten Datei (sparse file) kann diese Zahl größer als die wirkliche Dateilänge sein. Dünn besetzte Dateien können nicht über die normalen Operationen des Betriebssystems angelegt werden. Sie sind Ausnahmeerscheinungen und werden von einem Programm erzeugt, das Daten direkt im Verzeichnis einer Datei abändert. Wir werden sie einige Seiten später in diesem Kapitel kurz besprechen.

Anlegezeitpunkt (4 Bytes)

Die Bytes 25 bis 28 des Dateieintrags enthalten Datum und Uhrzeit des Zeitpunkts, an dem die Datei angelegt worden ist, abgespeichert als Binärzahl. Dieses Feld ist aus zwei kleineren Feldern zusammengesetzt. Die ersten zwei Bytes enthalten eine Binärzahl, die Jahr, Monat und Tag darstellt. Abb. 4.2 zeigt, wie diese Angaben dort gespeichert werden. Sie sehen, daß der Monat das unterste Bit von Byte 26 und die drei obersten Bits von Byte 25 belegt.

Das Feld für die Uhrzeit belegt die beiden nächsten Bytes. Byte 27 enthält die Minuten und hat immer Nullen in den Bits 7 und 6. Byte 28 enthält die Stunde; hier sind die Bits 5, 6 und 7 immer gleich Null. Das können Sie in Abb. 4.2 sehen.

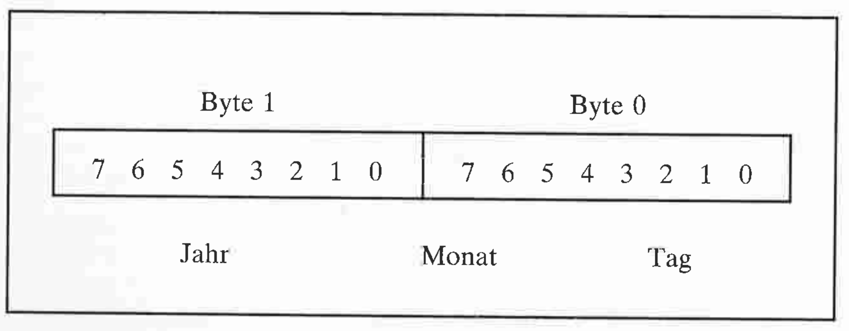


Abb. 4.1: Speichern des Datums im Feld für den Anlegezeitpunkt

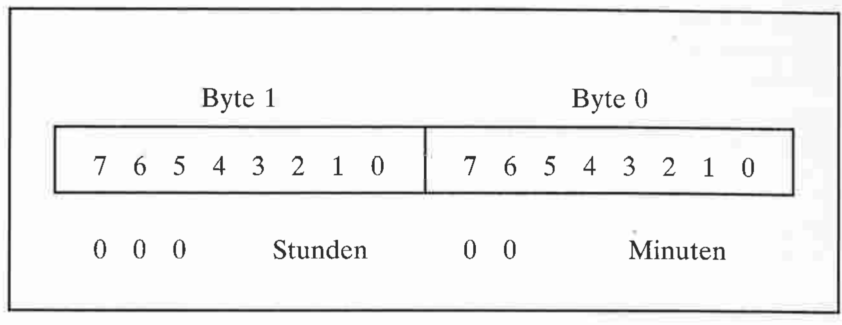


Abb. 4.2: Speichern der Uhrzeit im Feld für den Anlegezeitpunkt

Version (1 Byte)

Das nächste Byte enthält eine Binärzahl, die die Nummer der ProDOS-Version darstellt, mit der dieser Eintrag gemacht worden ist. Neuere Versionen von ProDOS können dieses Byte überprüfen und die notwendigen Anpassungen vornehmen.

Minimalversion (1 Byte)

Dieses Feld enthält die kleinste Nummer der ProDOS-Versionen, die diese Datei lesen können. Dadurch können ältere ProDOS-Versionen feststellen, ob sie von neueren Versionen angelegte Dateien lesen können.

Zugriff (1 Byte)

Dieses Dateieintragsfeld wird in erster Linie benutzt, um zu bestimmen, ob eine Datei schreibgeschützt ist. Es enthält auch noch ein Bit, das gesetzt werden kann, um anzuzeigen, ob eine Sicherungskopie der Datei angelegt worden ist. Wenn Sie Programme zum Erstellen von Sicherungskopien schreiben, die das Maschinensprache-Interface benutzen, sollten Sie sich vergewissern, daß Ihr Programm Bit 5 auf Null gesetzt hat und damit anzeigt, daß die Datei gesichert worden ist.

Aux_Typ (2 Bytes)

Programme können in diesem Feld zusätzliche Angaben über eine Datei speichern. Das ProDOS-Programm BASIC.SYSTEM verwendet das Feld mit diesem seltsamen Namen, um bei einer binären Datei die Ladeadresse oder bei einer Textdatei die Anzahl der Sätze (Records) festzuhalten.

Wenn Sie Maschinensprache- oder Assemblerprogramme schreiben, können Sie den Inhalt dieses Feldes für jeden beliebigen Zweck verwenden. Sie sind aber bei einer binären Datei unter Umständen nicht in der Lage, sie richtig laufen zu lassen oder zu laden, wenn Sie in diesem Feld keine Startadresse angeben, und Sie können bei einer Textdatei den Längenangaben im Verzeichnis nicht mehr trauen, wenn Sie die Angaben dort nicht selbst hineinschreiben. Speichern Sie diese Angaben nicht auf der Diskette, müssen Sie sie aufschreiben oder sich an sie erinnern, wenn Sie die Angaben später brauchen.

Letzte Modifikation (4 Bytes)

In diesem Feld ist aufgezeichnet, wann die Datei zum letztenmal geändert worden ist. Zum Speichern dieser Angaben als Binärzahl wird das gleiche Format wie im Feld für den Anlegezeitpunkt benutzt.

Kopf-Zeiger

Dieses Feld enthält die Adresse, die ProDOS benutzt, um auf das Verzeichnis zuzugreifen, zu dem diese Datei gehört. Kurz gesagt, ProDOS kann mit Hilfe dieses Feldes zum Herkunftsverzeichnis dieser Datei zurückgelangen.

UNTERVERZEICHNISDATEIEN

Der Zweck einer ProDOS-Unterverzeichnisdatei besteht darin, andere Dateien zu enthalten. Das bedeutet nicht, daß die wirklichen Daten einer Datei innerhalb dieser Unterverzeichnisdatei zu finden sind. Wie das Stammverzeichnis enthält ein Unterverzeichnis einen Dateieintrag für jede Datei, die mit diesem Unterverzeichnis angelegt worden ist. Wenn ProDOS eine Datei sucht, folgt es dem Pfadnamen vom Stammverzeichnis über jedes vorgeschaltete Unterverzeichnis, bis es die wirkliche Datei, die es sucht, gefunden hat. Auf jeder Stufe der Suche liest ProDOS eine Datei aus dem Verzeichnis oder Unterverzeichnis, um zu bestimmen, wohin es als nächstes gehen soll.

Folgende Schritte würden gemacht werden, folgte ProDOS dem Pfadnamen /PRODOS/VERZEICHNIS1/TESTDATEI.

1. ProDOS liest das Stammverzeichnis, um auf den Dateieintrag für VERZEICHNIS1 zu kommen.
2. Mit Hilfe der in Schritt 1 erhaltenen Angaben liest ProDOS die Unterverzeichnisdatei VERZEICHNIS1, um den Dateieintrag für TESTDATEI zu erhalten.

3. Mit Hilfe der in Schritt 2 erhaltenen Information ist ProDOS in der Lage, jede geforderte Operation an der Datei TESTDATEI auszuführen (vorausgesetzt, es handelt sich um einen legalen ProDOS-Befehl).

Jede Unterverzeichnisdatei besteht aus einem Satz für den Verzeichniskopf und je einem Satz für den Dateieintrag einer jeden in ihr enthaltenen Datei. Diese Dateieinträge haben das gleiche Format wie die Dateieinträge im Stammverzeichnis. Das Format des Kopfeintrags eines Unterverzeichnisses ist ähnlich aber nicht identisch mit dem Format des Kopfeintrags des Stammverzeichnisses. Es folgt eine Besprechung der Elemente des Kopfeintrags eines Unterverzeichnisses, aus der der Unterschied zum Kopfeintrag eines Stammverzeichnisses hervorgeht.

Speichertyp und Namenslänge sind aus zwei Feldern mit einer Länge von je vier Bits zusammengesetzt. Die oberen vier Bits sind auf $E (1110 in binärer Schreibweise) gesetzt und zeigen ProDOS an, daß es sich um den Schlüsselblock der Unterverzeichnisdatei handelt. Die unteren vier Bits enthalten die Namenslänge des Unterverzeichnisses.

Das Feld für den Dateinamen (15 Bytes lang) enthält den Namen der Unterverzeichnisdatei. Wie beim Feld für den Dateinamen im Kopfeintrag oder in einem Dateieintrag des Stammverzeichnisses muß das Feld für die Namenslänge auch hier bei jeder Namensänderung angepaßt werden.

Dann folgt ein reserviertes Feld (8 Bytes lang), das zur Zeit nicht benutzt wird und von Apple für zukünftige Erweiterungen reserviert wurde.

Das Feld für den Anlegezeitpunkt (4 Bytes lang) enthält Datum und Uhrzeit des Zeitpunkts, an dem das Unterverzeichnis von ProDOS initialisiert worden ist.

Das Versions-Byte enthält die Nummer der ProDOS-Version, mit der dieses Unterverzeichnis angelegt wurde.

Das Minimalversions-Byte legt die kleinste Versionsnummer fest, ab der alle ProDOS-Versionen auf dieses Unterverzeichnis zugreifen können.

Mit dem Zugriffs-Feld (1 Byte) läßt sich bestimmen, ob dieses Unterverzeichnis gegen Änderungen, Löschen oder Umbenennungen geschützt ist.

Das Eintragsfeld (1 Byte) besteht aus einer Binärzahl, die die Anzahl der Bytes eines jeden Eintrags in dieser Unterverzeichnisdatei angibt, eingeschlossen des Kopfeintrags. (Alle Einträge haben die gleiche Länge.)

Im Feld für die Einträge pro Block (1 Byte) ist die Anzahl der Einträge pro Block der Unterverzeichnisdatei in binärer Form gespeichert.

Das Dateizählfeld (2 Bytes lang) enthält die Anzahl der aktiven Dateieinträge in diesem Unterverzeichnis. Eine Datei heißt aktiv, wenn sie nicht den Speicherplatztyp 0 hat. Alle Standard- und Verzeichnisdateien gehören zu den aktiven Dateien.

Das Herkunftszeigerfeld ist zwei Bytes lang und enthält die Adresse des Blocks der Verzeichnisdatei, zu der diese Datei gehört. Das Herkunftsverzeichnis ist entweder das Stammverzeichnis oder ein anderes Unterverzeichnis.

Die Herkunftseintragszahl (1 Byte) enthält die Eintragsnummer dieses Unterverzeichnisses im Block, auf den der Herkunftszeiger zeigt. Wenn dieses Unterverzeichnis zum Beispiel die fünfte Datei im Herkunftsverzeichnis ist, enthält das Feld die Zahl 5.

Das Feld für die Länge des Herkunftseintrag (1 Byte) enthält die Länge eines jeden Eintrags im Herkunftsverzeichnis. (Alle Einträge haben die gleiche Länge.)

STANDARDDATEIEN

Standarddateien sind unter ProDOS alle Dateien, die keine Verzeichnisdateien sind. Alle Standarddateien fangen als nicht indizierte Dateien (seedling files) an und können sich zu einfach indizierten (sapling files) und weiter zu zweifach indizierten Dateien (tree files) ausdehnen. Eine nicht indizierte Datei ist natürlich der einfachste Dateityp; sie belegt genau 512 Bytes (1 Block) Speicherplatz.

Sobald sich eine Datei auf mehr als 512 Bytes ausdehnt, wird aus ihr eine einfach indizierte Datei. Während eine nicht indizierte Datei aus nur einem Block besteht, hat eine einfach indizierte Datei einen Indexblock und mindestens zwei Datenblöcke. Im Indexblock werden die Adressen der Datenblöcke, aus denen die Datei besteht, gespeichert. Daraus erklärt sich, warum es keine Standarddateien gibt, die genau zwei Blöcke auf einer Diskette belegen. Wenn alle Daten in einen Block hineinpassen, braucht die Datei nicht indiziert zu werden. Passen die Daten nicht in einen einzelnen Block, wird zu den zwei oder mehr Datenblöcken zusätzlich noch ein Indexblock angelegt.

Einfach indizierte Dateien können bis zu 256 Datenblöcke mit insgesamt 128K Daten erhalten. Wenn eine Datei noch größer wird, können die Adressen der Datenblöcke nicht mehr in einem einzelnen Indexblock gespeichert werden. Sie dehnt sich dann zu einer zweifach indizierten

Datei aus. Eine zweifach indizierte Datei hat einen Hauptindexblock, der auf maximal 128 Nebenindexblöcke zeigt, von denen jeder auf maximal 256 Datenblöcke zeigen kann. Das ergibt bis auf ein Byte eine maximale Dateigröße von 16 Megabytes. Das letzte Byte ist nicht erreichbar, weil ProDOS nicht zuläßt, daß die EOF-Marke (Dateiende) größer als 16 777 216 wird. ProDOS merkt sich automatisch, wenn Blöcke zugeteilt oder aufgelöst werden; Sie brauchen sich um diese Einzelheiten nicht zu kümmern.

Eine dünn besetzte Datei (sparse file) ist ein ungewöhnlicher Typ einer Standarddatei; es handelt sich um eine einfach oder zweifach indizierte Datei, die für eine bestimmte Aufgabe modifiziert worden ist. Dünn besetzte Dateien scheinen die ungewöhnliche Eigenschaft zu haben, daß man aus ihnen mehr Daten lesen kann, als physisch in der Datei auf der Diskette gespeichert sind.

Sie werden erzeugt, indem man die im Dateieintrag gespeicherten Angaben direkt ändert. Das können Sie tun, indem Sie bei einer bereits angelegten Datei die Markierung für das Dateiende (EOF) nach hinten verschieben, so daß ProDOS annimmt, die Datei sei größer, als sie in Wirklichkeit ist. Wenn Sie das bei einer nicht indizierten Datei tun und die Marke für das Dateiende so weit verschieben, daß ProDOS zusätzlichen Speicherplatz für einen Index- und einen Datenblock bereitstellen muß, macht es das erst, wenn Sie in diesen neuen Bereich Daten hineinschreiben. Wenn Sie das Dateiende einer nichtindizierten Datei so weit verschieben, daß fünf Datenblöcke erforderlich werden, und in den fünften Block Daten hineinschreiben, wird ProDOS diesen fünften Block anlegen und die Daten dort speichern. Der zweite, dritte und vierte Datenblock sind aber nicht angelegt worden, und ProDOS wird auf jeden Versuch, Daten aus diesen Blöcken zu lesen, mit der Ausgabe eines Null-Zeichens (ASCII 0) antworten. Das hat dann den Anschein, als würde man nicht existierende Daten lesen.

Diese Technik wird angewendet, wenn man auf einzelne Speicherstellen innerhalb einer Diskettendatei direkt zugreifen möchte, und wird nur für sehr erfahrene Programmierer empfohlen. Eine der Schwierigkeiten liegt in den unerwarteten Effekten, wenn man eine solche Datei mit einem Programm liest und sie wieder auf die Diskette schreibt. Die Ausgabedatei wird dann viel größer als die Eingabedatei, weil nämlich alle diese Null-Zeichen wirklich auf die Diskette geschrieben werden. Sie können jedoch die Datei-Dienstprogramme benutzen, um die Struktur einer dünn besetzten Datei beim Kopieren zu erhalten.

Dünn besetzte Dateien können in speziellen Situationen nützlich sein.

Wenn Sie ein außergewöhnlich großes Datenfeld unterhalten müssen, können dünn besetzte Dateien die einzige Möglichkeit sein, mit den Speicherplatzgrenzen einer Diskette fertig zu werden. Es werden nämlich nur die Blöcke angelegt, in die Sie auch wirklich hineinschreiben. Denken Sie daran, daß der ganze Block auf der Diskette angelegt wird, wenn Sie das erste Byte hineinschreiben.

In diesem Kapitel haben wir ein wichtiges Thema behandelt, das dem ganzen ProDOS-System zugrunde liegt – die Verzeichnis- und Dateistruktur, die ProDOS dazu verwendet, Daten zu speichern und zurückzuholen. Das Wichtigste, was Sie in diesem Kapitel gelernt haben, ist, wie die Struktur eines Verzeichnisses bei ProDOS angelegt ist. Diese Struktur ist der Schlüssel dazu, die Stärken von ProDOS wirklich auszunutzen. Sobald Sie sie verstanden haben und wissen, wie Präfixe und Pfadnamen zum Zugriff auf Dateien verwendet werden, sind Sie auf dem besten Weg, ProDOS zu beherrschen. Wenn Sie glauben, die Verzeichnisstruktur nicht verstanden zu haben, sollten Sie zuerst die Abschnitte über die Struktur einer unter ProDOS formatierten Diskette und die Organisation eines ProDOS-Datenträgers am Anfang dieses Kapitels wiederholen, bevor Sie weitermachen.

Kapitel 5
ProDOS-Befehle

In diesem Kapitel besprechen wir kurz die Befehlsfunktionen, die Ihnen ProDOS zur Verfügung stellt. Es gibt zwei Gruppen von ProDos-Befehlen: Die erste Gruppe enthält die DOS 3.3-Befehle, die auf den neuesten Stand gebracht worden sind; die zweite besteht aus neuen Befehlen, die es nur bei ProDOS gibt. Wir werden auch kurz die DOS-Befehle behandeln, über die ProDOS nicht mehr verfügt. Schließlich werden wir Änderungen bei den Applesoft-Befehlen besprechen, die beim Wechsel von DOS zu ProDOS erforderlich wurden.

Diese ProDOS-Befehle stehen vom BASIC aus zur Verfügung, entweder indem man sie direkt eintippt (im Direktmodus) oder indem man sie in ein BASIC-Programm einbettet. (Einige dieser Befehle funktionieren nicht im Direktmodus. In dem Fall werden wir darauf hinweisen.) Manche Funktionen stehen nicht nur über diese Befehle, sondern auch über die ProDOS-Dienstprogramme zur Verfügung, wie z. B. die Datei-Dienstprogramme – damit Sie es bequemer haben. Die meisten dieser Funktionen gibt es bei den Dienstprogrammen jedoch nicht, da sie in einer Menüstruktur nicht sinnvoll wären. Sie sind nur für Anwendungen innerhalb eines Programms vorgesehen.

AUF DEN NEUESTEN STAND GEBRACHTE BEFEHLE

Die folgenden Befehle gab es schon unter DOS 3.3, aber viele von ihnen sind unter ProDOS verbessert worden. DOS-Programme, in denen diese Befehle vorkommen, werden auch unter ProDOS laufen, aber ProDOS-Programme, die die neueren Eigenschaften dieser Befehle nutzen, laufen nicht unter DOS 3.3.

Der Befehl APPEND

Mit dem APPEND-Befehl können Sie Daten am Ende einer Datei anfügen. Unter DOS ging das nur bei sequentiellen Textdateien; unter Pro-

DOS können Sie damit Daten an jede beliebige Datei anfügen. Diesen Befehl gibt es nicht im Direktmodus. Da der APPEND-Befehl bei allen Dateitypen funktioniert, werden wir ihn in den Kapiteln 7 und 8 ausführlich besprechen.

Der Befehl BLOAD

Der BLOAD-Befehl wird in erster Linie dazu verwendet, binäre Dateien von einer Diskettendatei in den Speicher zu übertragen. Sie können ihn unter ProDOS jedoch bei jedem Dateityp dazu benutzen, um den Inhalt dieser Datei an eine bestimmte Stelle im RAM-Speicher des Apple zu laden. Unter ProDOS können Sie mit diesem Befehl auch einzelne Teile einer Datei in den Speicher laden. Eine detaillierte Besprechung des BLOAD-Befehls folgt in Kapitel 8.

Der Befehl BRUN

Der BRUN-Befehl überträgt den Inhalt einer binären Datei (BIN- Typ) in den Speicher und versucht, sie als Maschinensprache- Programm zu starten. Wie bei BLOAD können Sie unter ProDOS einen beliebigen Teil einer binären Datei mit BRUN laufen lassen. Mit dem BRUN-Befehl können Sie jedoch keine nichtbinären Dateien ausführen. Weil dieser Befehl in engem Zusammenhang mit binären Dateien steht, werden wir ihn in Kapitel 8 im Detail besprechen.

Der Befehl BSAVE

Der BSAVE-Befehl überträgt Daten vom Speicher in eine Diskettendatei. Diese Daten werden binär gespeichert. Binäre Dateien werden im allgemeinen zum Speichern von Maschinensprache- Programmen, für Grafik und zum schnellen Laden und Abspeichern des Speicherinhalts verwendet. Sie können damit zum Beispiel die binäre Darstellung eines Grafikbildes aus dem Speicher auf eine Diskettendatei übertragen. Auf diese Weise können Sie manchmal den Code eines Programms verkürzen, weil Sie die Zeilen weglassen können, die zur Erzeugung des Grafikbildes erforderlich sind. Diese können Sie dann durch einen einzelnen BLOAD-Befehl ersetzen. Für weitere Informationen zum BSAVE-Befehl im Zusammenhang mit binären Daten verweisen wir auf Kapitel 8.

Der Befehl CATALOG

Der CATALOG-Befehl unter DOS 3.3 gibt Ihnen Typ, Größe und Namen aller Dateien auf einer Diskette an. Unter ProDOS zeigt Ihnen CATALOG volle 80 Spalten Information über die Dateien in einer

Stammverzeichnis- oder Unterverzeichnisdatei. Wenn Sie keine 80- Zeichen-Karte haben, sollten Sie statt dessen den CAT-Befehl (ein neuer ProDOS-Befehl) verwenden; denn die 80 Zeichen einer vom ProDOS-CATALOG-Befehl ausgegebenen Zeile werden auf einem Schirm mit nur 40 Zeichen pro Zeile auf zwei Zeilen gefaltet und sind deshalb nur schwer lesbar.

Bei beiden Befehlen erhalten Sie die gleichen Angaben über ein Verzeichnis. Dazu gehören der Dateiname, der Typ, die Länge und das Datum der letzten Änderung einer jeden Datei. Zusätzlich zeigt Ihnen der CATALOG-Befehl noch das Datum, an dem die Datei angelegt worden ist, das logische Ende der Datei und entweder die Länge der Datei in Sätzen (wenn es sich um eine Textdatei mit wahlfreiem Zugriff handelt) oder die letzte Ladeadresse (wenn es sich um eine binäre Datei handelt).

Sowohl der CAT- als auch der CATALOG-Befehl zeigen Ihnen den Inhalt einer einzelnen Verzeichnisdatei. Wenn Sie keinen Diskettennamen angeben, erscheint der Inhalt der Verzeichnisdatei, auf die das Präfix zeigt. Wollen Sie den Inhalt eines anderen Verzeichnisses sehen, dann müssen Sie hinter dem CAT- oder dem CATALOG-Befehl einen Pfadnamen angeben.

Der Befehl CLOSE

Der CLOSE-Befehl wird dazu verwendet, um Dateien, die mit dem OPEN-Befehl (der wenig später besprochen wird) geöffnet worden sind, wieder zu schließen. Durch das Schließen der Datei wird sichergestellt, daß alles, was in die Dateipuffer geschrieben worden ist, auch auf die Diskette übertragen wird. ProDOS legt die Daten vorläufig in einem Dateipuffer ab, um die Ausführungsgeschwindigkeit eines Programms zu vergrößern. Wenn ein solches Programm endet, ohne daß die Dateien, die während seiner Ausführung geöffnet wurden, auch wieder geschlossen worden sind, können die Daten in diesen Dateipuffern verlorengehen.

Falls ein Programm wegen eines Fehlers vorzeitig abgebrochen wurde und einige Dateien deshalb nicht geschlossen worden sind, können Sie den CLOSE-Befehl nachträglich im Direktmodus von BASIC über die Tastatur eingeben. Solange Sie kein anderes Programm ausgeführt oder den Speicher gelöscht haben, können Sie so die Daten in den Dateipuffern noch auf die Diskette retten. Der CLOSE-Befehl sorgt auch dafür, daß die von ProDOS angelegten Dateipuffer in der richtigen Reihenfolge wieder freigegeben werden. Wenn Sie keinen Pfad- oder Dateinamen im CLOSE-Befehl angegeben haben, werden alle offenen Dateien geschlossen.

Bei ProDOS müssen Sie mit dem CLOSE-Befehl alle offene Dateien wieder schließen, wenn Sie nicht den Verlust von Daten riskieren wollen. Das ist ein wesentlicher Unterschied zwischen DOS und ProDOS, über den Sie sich immer im klaren sein müssen, wenn Sie DOS-Programme nach ProDOS konvertieren.

Der Befehl DELETE

Der DELETE-Befehl löscht die Datei oder die Dateien, die Sie angeben, von der Diskette. Wenn Sie eine Datei einmal gelöscht haben, gibt es keine Möglichkeit, sie zurückzuholen. Bei einer geschützten Datei muß der Schreibschutz entfernt werden, bevor sie gelöscht werden kann. Eine Verzeichnisdatei muß leer sein, bevor ProDOS zuläßt, daß sie gelöscht wird. Das ist der einzige Unterschied zwischen DOS und ProDOS bei diesem Befehl. Wenn Sie eine Verzeichnisdatei löschen wollen, müssen Sie zuerst jede einzelne Datei in diesem Verzeichnis löschen. Erst dann können Sie die Verzeichnisdatei selbst löschen. Eine Datei, die mehrere Stufen innerhalb des gegenwärtigen Verzeichnisses liegt, können Sie löschen, indem Sie einen Pfadnamen angeben, der mit dem Namen der Datei endet, die Sie löschen wollen.

Der Befehl EXEC

Der EXEC-Befehl veranlaßt den Apple, die Standardeingabe nicht von der Tastatur, sondern von einer Diskettendatei zu lesen. Diese Diskettendatei kann eine Folge von Anweisungen enthalten, wie zum Beispiel RUN PROGRAMM1 oder DELETE MEINEDATEI, oder sie kann aus einer Ansammlung von Daten bestehen. Daten und Anweisungen können auch kombiniert werden. Dieser Befehl ist unverändert von DOS 3.3 übernommen worden, außer daß man einen ProDOS-Pfadnamen angeben muß, um eine Datei zu spezifizieren.

Der Befehl IN#

Der IN#-Befehl leitet die Standardeingabe Ihres Apple um. Normalerweise erwartet der Apple Eingaben über die Tastatur. Sie können den IN#-Befehl dazu verwenden, die Standardeingabe von der Tastatur auf ein beliebiges Gerät umzuleiten, das an einem der acht Erweiterungssteckplätze mit den Nummern 0 bis 7 angeschlossen ist. Wenn Sie bei diesem Befehl eine Zahl angeben, die nicht im Bereich 0 bis 7 liegt, erscheint eine Fehlermeldung.

Mit diesem Befehl können Sie Ihren Apple über ein externes Gerät steuern, wie zum Beispiel über ein zweites Terminal oder über ein Disketten-

laufwerk. Wenn es sich um ein Diskettenlaufwerk handelt, aktiviert der IN#-Befehl ein Maschinensprache-Programm auf der Laufwerkkontrollkarte und lädt von der Diskette, die sich gerade im entspechenden Laufwerk befindet. Das Programm schaut zuerst im ersten an die Karte angeschlossenen Laufwerk (Drive 1) nach.

Auf einem Apple IIc gibt es keine Erweiterungssteckplätze. Es gibt nur die Eingänge auf der Rückseite des Computers und den internen Kreislauf. ProDOS erkennt die beiden seriellen Eingänge als Steckplatz 1 und 2. Die eingebaute 80-Zeichen-Karte wird als Steckplatz 3 behandelt, und Steckplatz 4 ist der Eingang für den Joystick auf der Rückseite der Maschine. Das eingebaute Laufwerk belegt Steckplatz 6 und das externe Laufwerk Steckplatz 7. Diese Interpretation des Eingangs für das externe Laufwerk ist eine Ausnahme zu den in Kapitel 3 beschriebenen Beziehungen zwischen den Eingängen beim IIc und den Steckplätzen beim IIe.

Normalerweise müßte der IIc den Eingang für das externe Laufwerk als Steckplatz 6, Laufwerk 2 interpretieren. Apple hat das geändert, um Ihnen eine Möglichkeit zu geben, den Computer über das externe Laufwerk zu starten, wenn das eingebaute Laufwerk nicht richtig funktioniert. Sie können Ihr System neu starten, indem Sie den IN#-Befehl mit der Nummer eines Steckplatzes eingeben, an den ein Laufwerk angeschlossen ist. Dann versucht der Apple das Betriebssystem auf der Diskette in dem entsprechenden Laufwerk zu laden. Jedes Programm, das sich im Speicher befindet, wird dann entfernt. Wenn Sie den Befehl

```
IN#0
```

eingeben, liest der Apple die Eingabe wieder von der Tastatur.

Mit dem IN#-Befehl kann man auch ein Maschinensprache-Programm aufrufen. Wenn Sie die Adressenoption zum Bestimmen einer Startadresse beim IN#-Befehl verwenden, können Sie Ihren Apple dazu veranlassen, die Routine auszuführen, die bei dieser Adresse beginnt. Das Thema der Adreßoptionen wird in Kapitel 8 ausführlich behandelt.

Der Befehl LOAD

Der LOAD-Befehl überträgt eine BASIC-Programmdatei von der Diskette in den Speicher. Dieser Befehl wird im allgemeinen dann verwendet, wenn man ein BASIC-Programm überprüfen oder ändern möchte, ohne es laufen zu lassen. Wenn Sie das Programm nur laufen lassen wollen, genügt der RUN-Befehl, dann wird die Programmdatei automatisch geladen. Beim Laden eines Programms in den Speicher wird der alte Spei-

cherinhalt gelöscht. Wenn Sie Variablenwerte des alten Programms bewahren wollen, können Sie sie mit dem STORE-Befehl in einer Datei speichern oder den CHAIN- Befehl verwenden.

Die BASIC-Programmdatei auf der Diskette wird durch den LOAD-Befehl in keiner Weise verändert; es wird nur eine Kopie von ihr im Speicher abgelegt. Wenn Sie das Programm im Speicher geändert haben, brauchen Sie den SAVE-Befehl, um diese Änderungen auf Diskette zu speichern.

Dieser Befehl hat sich außer in der Verwendung eines Pfadnamens bei der Angabe einer Datei gegenüber DOS 3.3 nicht geändert.

Der Befehl LOCK

Mit dem LOCK-Befehl können Sie eine Datei mit einem Schreibschutz versehen. Es ist nicht möglich, eine geschützte Datei umzubenennen, sie zu löschen oder irgenwelche Daten in der Datei zu verändern. Alle geschützten Dateien sind bei der CATALOG- Anzeige vor ihrem Namen mit einem Stern gekennzeichnet. Sie können nur nicht das Stammverzeichnis einer Diskette mit dem LOCK-Befehl schreibschützen, das erreichen Sie aber durch Überkleben der Schreibschutzkerbe auf der Diskette. Unterverzeichnisdateien können jedoch mit dem LOCK-Befehl schreibgeschützt werden. Bis auf die Verwendung eines ProDOS- Pfadnamens ist dieser Befehl unverändert von DOS 3.3 übernommen worden.

Der Befehl OPEN

Der OPEN-Befehl legt einen Dateipuffer zum Lesen und Schreiben in eine Datei an. Er bereitet auch das Beschreiben und Lesen dieser Datei, angefangen mit dem ersten Byte der Datei, vor. Maximal acht Dateien dürfen gleichzeitig mit dem OPEN-Befehl geöffnet sein. (Unter DOS wurde die maximale Anzahl der Dateien, die gleichzeitig geöffnet sein durften, durch den MAXFILES-Befehl festgelegt. Das waren normalerweise 3, aber die Anzahl konnte bis auf 16 erhöht werden.) Jeder geöffneten Datei wird ein Dateipuffer von 1024 Bytes (512 Bytes für eine Verzeichnisdatei) zugewiesen. Jede geöffnete Datei muß später mit dem CLOSE-Befehl geschlossen werden. Der OPEN-Befehl kann nicht im Direktmodus verwendet werden.

Dieser Befehl hat unter ProDOS keine zusätzlichen Eigenschaften, nur wird jetzt bei der Ausführung des OPEN-Befehls der Dateipuffer angelegt. Unter DOS wurde einer Datei ein Standardpuffer zugewiesen, der

mit dem MAXFILES-Befehl geändert werden konnte. Den MAXFILES-Befehl gibt es unter ProDOS nicht mehr. Der OPEN-Befehl wird in Kapitel 7 noch ausführlicher behandelt.

Der Befehl POSITION

Der POSITION-Befehl ist unter ProDOS nicht erforderlich; er wurde beibehalten, um zu DOS 3.3 kompatibel zu bleiben. Da nun einmal an ihm festgehalten wurde, haben ihn die Designer von ProDOS um eine neue Option erweitert, um ihn an die Dateibehandlung durch ProDOS anzupassen. Mit diesem Befehl kann man sich die Daten in jedem Bereich oder in jedem Byte innerhalb einer Datei anschauen. Der POSITION-Befehl kann nicht im Direktmodus verwendet werden. Für weitere Informationen, wann und wie dieser Befehl zu benutzen ist, verweisen wir auf Kapitel 7.

Der Befehl PR#

Der PR#-Befehl arbeitet ähnlich wie der IN#-Befehl, außer daß er nicht kontrolliert, woher die Eingabe kommt, sondern, wohin die Ausgabe geht. Normalerweise erfolgt die Ausgabe Ihres Apple über den Monitor. Mit dem PR#-Befehl können Sie die Ausgabe auf ein Gerät umleiten, das an einen der Steckplätze des Apple angeschlossen ist. Wenn Sie zum Beispiel an Steckplatz 1 einen Drucker angeschlossen haben, erfolgt nach dem Eingeben von

```
PR#1
```

die gesamte Ausgabe über diesen Drucker. Nach PR#0 wird wieder über den Monitor ausgegeben. Mit dem PR#-Befehl können Sie auch das System neu starten. Wenn Sie zum Beispiel eine Laufwerkkontrollkarte in Steckplatz 6 haben, aktiviert der Befehl PR#6 das Maschinensprache-Programm auf dieser Karte, genau wie der Befehl IN#6. Dann wird das System neu gestartet und das Betriebsystem geladen, das sich auf der Diskette in dem an diese Karte angeschlossenen Laufwerk befindet. Seien Sie vorsichtig mit diesem Befehl, denn er entfernt jedes Programm aus dem Speicher.

Wie schon vorher erwähnt, hat der Apple 8 Steckplätze, die von 0 bis 7 numeriert sind. Das sind die einzigen Zahlen, die beim PR#-Befehl eingesetzt werden dürfen. Jede andere Zahl verursacht eine Fehlermeldung. Auf einem Apple IIc erkennt ProDOS die beiden seriellen Eingänge als Steckplatz 1 und 2. Die eingebaute 80-Zeichen-Karte wird als Steckplatz

3 behandelt, Steckplatz 4 ist der Ausgang für den Joystick. Das eingebaute Laufwerk entspricht Steckplatz 6 und das externe Laufwerk Steckplatz 7.

Wenn Sie in Ihrem Apple IIe eine 80-Zeichen-Karte haben und diese gerade im Gebrauch ist, müssen Sie sie abschalten, bevor Sie einen weiteren PR#-Befehl benutzen können. Das erreichen Sie, indem Sie zuerst die Escape-Taste und danach gleichzeitig die Control- und die Q-Taste drücken. Der Apple IIc hat eine eingebaute 80-Zeichen-Karte, die mit PR#3 eingeschaltet wird.

Mit dem PR#-Befehl können Sie auch ein Maschinensprache-Programm aufrufen. Das geht mit der Adressenoption, die der einzige neue Bestandteil von PR# unter ProDOS ist. Wenn Sie mit der Adressenoption die Startadresse beim PR#-Befehl festlegen, veranlassen Sie Ihren Apple, das an dieser Stelle beginnende Maschinensprache-Programm auszuführen. Das Gebiet der Adressenoptionen wird in Kapitel 8 ausführlich behandelt werden.

Der Befehl READ

Der READ-Befehl sagt ProDOS, welche Datei es für die nächste INPUT- oder GET-Anweisung verwenden soll. Sie müssen einen READ-Befehl eingeben, bevor Sie Daten aus einer Datei lesen können, und die Datei muß geöffnet sein. Der READ-Befehl bleibt gültig, bis ein anderer ProDOS-Befehl eingegeben wird. Den READ-Befehl gibt es nicht im Direktmodus.

Dieser Befehl wird in erster Linie bei Textdateien angewendet, er wird in Kapitel 7 genauer besprochen. Der READ-Befehl hat einen neuen Parameter, mit dem Sie seine Leseposition in einer Datei nach Feldern oder Bytes festlegen können. Das macht den POSITION-Befehl von DOS überflüssig.

Der Befehl RENAME

Mit dem RENAME-Befehl können Sie den Namen einer Datei ändern. Wenn Sie diesen Befehl verwenden wollen, tippen Sie RENAME gefolgt von dem Pfadnamen, der mit der Datei endet, die Sie umbenennen wollen, und dann den gleichen Pfadnamen, der jetzt aber mit dem neuen Dateinamen endet. Die beiden Pfadnamen müssen durch ein Komma getrennt werden. Sie können mit diesem Befehl keine schreibgeschützten Dateien umbenennen und auch keine Datei in ein anderes Verzeichnis übertragen. Dieser Befehl unterscheidet sich von der DOS-Version nur

durch die Angabe eines Pfadnamens, der bei ProDOS zum Lokalisieren einer Datei erforderlich ist.

Der Befehl RUN

Der RUN-Befehl veranlaßt ProDOS, ein Applesoft-Programm auszuführen. Wenn Sie bei diesem Befehl keinen Dateinamen angeben, versucht ProDOS, das Applesoft-Programm im Speicher laufen zu lassen. Befindet sich dort kein Applesoft-Programm, passiert gar nichts. Wenn Sie beim RUN-Befehl einen Namen angeben, wird ProDOS die Datei mit diesem Namen suchen, in den Speicher laden und mit der Ausführung des Programms beginnen. Nachdem Sie ein Programm so geladen haben, brauchen Sie nur RUN einzutippen und die Return-Taste zu drücken, um das Programm noch einmal laufen zu lassen (solange Sie kein anderes Programm geladen haben).

Es gibt zwei Unterschiede bei den RUN-Befehlen unter DOS und ProDOS. Als erstes können Sie bei ProDOS natürlich einen Pfadnamen hinter dem RUN-Befehl eingeben. Als zweites können Sie eine Zeilennummer festlegen, bei welcher mit der Ausführung des Programms begonnen werden soll. Das geschieht, indem Sie hinter dem Pfadnamen ein Komma, ein @-Symbol und eine Zeilennummer eingeben. Die Anweisung

```
RUN TEST.PROGRAMM,@1000
```

würde das Programm TEST.PROGRAMM ab Zeile 1000 ausführen. Wenn Sie keine Zeilenmummer angeben, beginnt die Ausführung des Programms bei der kleinsten Zeilennummer.

Der Befehl SAVE

Der SAVE-Befehl schreibt das BASIC-Programm, das sich gerade im Speicher befindet, in die Diskettendatei, die Sie angeben. Wenn Sie keinen Dateinamen eingeben, bekommen Sie eine „Syntax Error"-Fehlermeldung. Wenn die Datei neu ist, wird sie automatisch als BASIC-Datei angelegt. Eine bereits existierende Datei mit diesem Namen kann leider nicht schreibgeschützt werden. Das geht beim Dateityp der BASIC-Programme nicht. Wenn Sie es versuchen, werden Sie eine Fehlermeldung erhalten. Natürlich können Sie mit dem SAVE-Befehl eine Sicherungskopie des Programms anlegen, indem Sie einen anderen Dateinamen als den bereits existierenden angeben.

Seien Sie also sehr vorsichtig mit dem SAVE-Befehl, wenn Sie einen Dateinamen angeben, den es schon gibt. Die bereits existierende Datei wird nämlich überschrieben. Arbeiten Sie mit mehreren verschiedenen Programmdateien, dann achten Sie darauf, daß Sie ein Programm nicht unter falschem Namen abspeichern. Wenn eine Datei geändert oder gelöscht worden ist, können Sie sie nicht mehr zurückbekommen, außer Sie haben eine Sicherungskopie. Dieser Befehl ist unter DOS und ProDOS gleich, außer daß unter ProDOS ein Pfadname angegeben werden muß.

Der Befehl UNLOCK

Mit dem UNLOCK-Befehl können Sie den Schreibschutz einer Datei wieder entfernen. Sie können dann die vorher geschützte Datei wieder löschen, umbenennen oder ändern. Das geht nicht bei einer Stammverzeichnisdatei, weil sie mit dem LOCK-Befehl nicht schreibgeschützt werden kann. Wenn Sie zu diesem Zweck die Schreibschutzkerbe der Diskette mit einer Lasche überklebt haben, können Sie die Lasche natürlich wieder abnehmen, um den Schreibschutz des Stammverzeichnisses zu entfernen. Sie können den UNLOCK-Befehl auf jede geschützte Datei anwenden, eingeschlossen Unterverzeichnisdateien. Bis auf die erforderliche Angabe eines Pfadnamens ist dieser Befehl unverändert von DOS 3.3 übernommen worden.

Der Befehl WRITE

Der WRITE-Befehl muß erfolgen, bevor Sie mit dem PRINT-Befehl Daten in eine Diskettendatei schreiben können. Mit dem WRITE-Befehl sagen Sie ProDOS, in welche Datei die Zeichen geschrieben werden sollen. Diese Datei muß vor dem WRITE-Befehl geöffnet worden sein. Die Wirkung von WRITE dauert bis zum nächsten ProDOS-Befehl an. Den WRITE-Befehl gibt es nicht im Direktmodus.

Dieser Befehl wird gewöhnlich den Textdateien zugeordnet, wir werden ihn deshalb in Kapitel 7 ausführlich besprechen. Der WRITE-Befehl hat die gleichen zusätzlichen Parameter wie der READ-Befehl erhalten, so daß auch hier der POSITION-Befehl überflüssig wird.

NEUE BEFEHLE

Die folgenden Befehle gab es nicht unter DOS 3.3. Sie sind hinzugefügt worden, um das neue Betriebssystem mächtiger zu machen.

Der Befehl CAT

Der CAT-Befehl ist eine abgekürzte Form des vorher beschriebenen CATALOG-Befehls. Der CAT-Befehl gibt nur 40 Spalten Informationen über eine Verzeichnisdatei aus, während es bei CATALOG 80 Spalten sind. Dieser Befehl gibt den Namen, den Typ, die Länge und das Datum der letzten Änderung einer jeden in dem Verzeichnis enthaltenen Datei aus. Das sind die gleichen Daten, die Sie zu sehen bekommen, wenn Sie sich ein Verzeichnis mit den Datei-Dienstprogrammen auflisten lassen. Die beiden Befehle CAT und CATALOG sind dazu da, daß Sie sich auch während der Arbeit im BASIC-Modus ein Verzeichnis anschauen können.

Der Befehl CHAIN

Mit dem CHAIN-Befehl können Sie ein Programm von einem anderen Programm aus starten. Es gibt einige Vorteile, wenn man den CHAIN-Befehl anstelle des RUN-Befehls verwendet. Der größte Vorteil besteht darin, daß der CHAIN-Befehl nicht die Werte aller Variablen löscht, wenn er das nächste Programm lädt und ausführt. Wenn Sie diesen Befehl beim Programmieren verwenden, können Sie ein sehr großes und komplexes Programm in kleine Einheiten aufteilen und dann diese kleinen Programme mit dem CHAIN-Befehl verketten, um sie so übersichtlicher zu machen. Da alle Variablenwerte beim Übergang von einem Teilprogramm zum nächsten erhalten bleiben, sparen Sie Zeit, weil diese Werte nicht von einem Programm auf einer Diskette zwischengespeichert und von einem anderen wieder eingelesen werden müssen.

Der Befehl CREATE

Mit dem CREATE-Befehl können Sie eine Datei eines beliebigen Typs erzeugen, aber er wird hauptsächlich zum Anlegen von Verzeichnisdateien verwendet. Deshalb ist er auch bei ProDOS hinzugefügt worden. BASIC-, Text- und Binärdateien werden automatisch durch die Befehle SAVE, OPEN oder BSAVE angelegt. Textdateien können auch mit dem APPEND-Befehl und Variablendateien mit dem STORE-Befehl erzeugt werden. Beim Anlegen einer Datei wird in der entsprechenden Verzeichnisdatei ein Eintrag gemacht, damit ProDOS diese Datei finden kann. Mit den Datei-Dienstprogrammen können Sie diese Dateien von einem Verzeichnis in ein anderes kopieren.

Der Befehl FLUSH

Der FLUSH-Befehl veranlaßt ProDOS, den Inhalt des angegebenen

Dateipuffers auf die Diskette zu schreiben. Bei häufiger Verwendung dieses Befehl gehen Sie sicher, daß Sie keine Daten durch einen Systemfehler oder durch einen zufälligen Abbruch des Programms verlieren. Die Ausführung des Programms wird durch den Befehl allerdings beträchtlich langsamer.

Ein Grund dafür, daß bei ProDOS das Arbeiten mit Disketten so viel schneller als bei DOS ist, besteht darin, daß ProDOS wartet, bis ein Dateipuffer voll oder fast voll ist, bevor es den Inhalt auf die Diskette überträgt. Durch weniger häufiges Benutzen des Diskettenlaufwerks wird die Ausführungsgeschwindigkeit eines Programms beträchtlich größer. Der Nachteil dieser Methode ist: Bei Auftreten eines Systemfehlers haben Sie die Daten noch nicht auf eine Diskette gerettet. Sie sind verloren und müssen neu berechnet werden. Der FLUSH-Befehl veranlaßt ProDOS, den Dateipuffer früher zu retten, verlangsamt aber das Programm durch häufigeren Diskettenzugriff.

Der Befehl FRE

Mit dem FRE-Befehl wird der Speicherplatz nicht mehr benötigter Variablen zur Wiederverwendung neu aufbereitet. Er arbeitet genauso wie der FRE-Befehl im Applesoft-BASIC, aber die ProDOS- Version ist schneller. Das liegt an einer kleinen Korrektur des Applesoft-BASIC, die bei ProDOS gemacht wurde.

Der Befehl PREFIX

Der PREFIX-Befehl arbeitet genauso wie die SET PRODOS PREFIX-Option bei den Datei-Dienstprogrammen. Nach Eintippen von PREFIX und Drücken der Return-Taste zeigt ProDOS das zur Zeit verwendete Präfix an. Wenn Sie PREFIX, gefolgt von einem Pfadnamen, eintippen, wird das alte Präfix durch diesen Pfadnamen ersetzt. Geben Sie einen ungültigen Pfadnamen ein, erscheint die Fehlermeldung PATH NOT FOUND, und Sie müssen es noch einmal versuchen.

Der Befehl RESTORE

Der RESTORE-Befehl lädt den Inhalt einer Variablendatei in den Speicher. Die Werte der Variablendatei sind dort vorher mit dem STORE-Befehl gespeichert worden. Diese Variablen werden zu den bisherigen Variablen im Programm hinzugefügt. Wenn der Name einer Dateivariablen im Speicher schon vorkommt, wird der Wert der Variablen im Speicher durch den Wert der Variablen aus der Datei ersetzt.

Der Befehl STORE

Der STORE-Befehl schreibt den Namen und die Werte aller durch ein BASIC-Programm im Speicher definierten Variablen in eine Variablendatei (Typ VAR). Diese Variablen können mit dem RESTORE- Befehl an ein anderes BASIC-Programm übergeben werden.

Der Befehl – (Strich)

Der aus dem einzelnen Zeichen – bestehende Befehl heißt Strich-Befehl (dash command). Mit diesem Befehl können Sie sowohl BASIC-, Binär-, EXEC- als auch Systemprogramme laufen lassen. Sie tippen einfach einen Strich, gefolgt von dem Namen des Programms, ein. Allerdings stehen Ihnen nicht die Zusatzoptionen wie bei den Befehlen RUN, BRUN und EXEC zur Verfügung.

NICHT MEHR VERWENDETE BEFEHLE

Die folgenden DOS-Befehle gibt es unter ProDOS nicht mehr. Vier dieser Befehle (FP, INIT, INT und MON) führen unter ProDOS zu einem Syntaxfehler. In diesem Fall müssen Sie diese Befehle aus einem Programm entfernen, bevor Sie es unter ProDOS laufen lassen können.

Der Befehl FP

Der FP-Befehl wurde unter DOS zum Umschalten von Integer-BASIC nach Applesoft-BASIC verwendet. ProDOS unterstützt nur Applesoft-BASIC im ROM und hat keine Verwendung für diesen Befehl. ProDOS selbst benutzt den Speicherbereich, den in den RAM-Speicher geladenes Applesoft- oder Integer-BASIC belegen würde, so daß diese Möglichkeit entfällt.

Der Befehl INIT

Mit dem INIT-Befehl wurden bei DOS 3.3 Disketten formatiert. Das wird unter ProDOS vom Menü der Datei-Dienstprogramme aus gemacht. Sie können also keine Disketten mehr von einem Programm aus formatieren. Deshalb müssen Sie jederzeit genügend formatierte Leerdisketten zur Hand haben.

Unter INIT konnte ein Begrüßungsprogramm jeden beliebigen Namen haben; unter ProDOS muß es STARTUP heißen. Unter DOS konnten Begrüßungsprogramme nur in BASIC geschrieben werden; unter ProDOS können Sie ein BASIC-, ein Maschinensprache- oder ein EXEC-Programm als STARTUP-Programm verwenden.

Der Befehl INT

Mit dem INT-Befehl wurde unter DOS von Applesoft-BASIC nach Integer-BASIC umgeschaltet. Da unter ProDOS Integer-BASIC nicht mehr geladen werden kann, ist dieser Befehl überflüssig geworden.

Der Befehl MAXFILES

Mit dem MAXFILES-Befehl wurde unter ProDOS die maximale Anzahl gleichzeitig geöffneter Dateien festgelegt. Diese Zahl war standardmäßig auf 3 gesetzt und konnte mit dem MAXFILES-Befehl bis auf 16 erhöht werden. Unter ProDOS können in einem Programm maximal acht Dateien gleichzeitig geöffnet sein. ProDOS erlaubt keine Änderung dieser Zahl und ignoriert deshalb den MAXFILES-Befehl einfach, wenn es auf ihn trifft.

ProDOS weist jeder Standarddatei automatisch einen 1024 Bytes großen Puffer zu, wenn sie geöffnet wird. Einer Verzeichnisdatei wird ein 512 Bytes großer Puffer zugewiesen. Das beeinflußt die maximale Größe eines BASIC-Programms. Auf einem Apple II mit 64K Speicherplatz wird die maximale Größe eines BASIC-Programms durch die Formel

maximale Größe = 40960 − (1024 * maximale Anzahl der offenen Dateien)

berechnet.

Der Befehl MON

Mit dem MON-Befehl konnten Sie sich unter DOS die gesamte Disketteneingabe und -ausgabe auf dem Bildschirm anzeigen lassen, ohne sie ausdrucken zu müssen. Dieser Befehl wird von ProDOS nicht mehr unterstützt.

Der Befehl NOMON

Der NOMON-Befehl schaltete die Änderungen des MON-Befehls wieder ab. ProDOS ignoriert diesen Befehl, ohne daß ein Fehler entsteht.

ÄNDERUNGEN AM APPLESOFT

Wie in Kapitel 1 erwähnt, läuft Applesoft-BASIC unter ProDOS geringfügig langsamer als unter DOS 3.3. Das liegt daran, daß ProDOS jede Zeile eines BASIC-Programms daraufhin untersucht, ob sie eine PRINT-

Anweisung, gefolgt von Control-D als erstem auszudruckendem Zeichen, enthält. ProDOS verwendet dieses Zeichen, um zu erkennen, was mit dem Befehl gemeint war. DOS 3.3 hielt auch nach dem Control-D Ausschau, aber es überprüfte nicht jeden Applesoft-Befehl während seiner Ausführung. DOS schaute nur auf Zeichen, die vom Programm ausgegeben wurden. Das ging zwar schneller, aber es bedeutete, daß jedem Control-D-Zeichen ein Return-Zeichen (ASCII 13) vorangehen mußte, damit es als solches erkannt wurde. Apple entschied sich dafür, die Verwendung von ProDOS-Befehlen innerhalb eines BASIC-Programms etwas einfacher zu machen und und brachte dafür ein kleines Opfer in der Ausführungsgeschwindigkeit. Weil Diskettenoperationen unter ProDOS schneller sind als unter DOS, werden Ihre Programme möglicherweise trotzdem unter ProDOS schneller laufen.

Bei der Entwicklung von DOS zu ProDOS hielt es Apple für nötig, Änderungen bei einigen Applesoft-Befehlen vorzunehmen. Zehn Befehle sind davon betroffen. Es folgt eine kurze Dikussion der Änderungen und Einschränkungen bei diesen Befehlen.

Der HIMEM-Befehl ist davon betroffen, weil ProDOS den HIMEM-Wert dazu verwendet, den Bereich des I/O-Puffers einer neu geöffneten Datei festzulegen. ProDOS teilt den Speicher in 256 Byte große Segmente ein. Deshalb muß der HIMEM-Wert immer ein Vielfaches von 256 sein. Wenn Sie mit Hexadezimalzahlen arbeiten, heißt das, daß der HIMEM-Wert immer ein Vielfaches von $100 sein muß.

Die Befehle HGR, HGR2 und TEXT haben alle mit dem Grafikbereich des Speichers Ihres Apple II zu tun. Da ProDOS mehr Speicherplatz als DOS belegt, hat ProDOS diesen Speicherbereich in den normalen Speicherbereich von Applesoft-Programmen aufgenommen. Wenn Sie die Applesoft-Befehle HGR und HGR2 innerhalb eines Programms verwenden, wird der Inhalt dieses Speicherbereichs entfernt. Dieser Bereich bleibt dann für Grafik reserviert, bis der TEXT-Befehl ausgeführt oder das System neu gestartet wird.

Auch der Applesoft-Befehl INPUT ist geändert worden. Unter DOS würde die INPUT-Anweisung alle eingetippten Zeichen hinter einem Komma oder einem Semikolon ignorieren. Würden Sie beispielsweise die Zeichenkette ABCD,EFGH bei einer INPUT-Anweisung eingeben, erhielten Sie die Meldung EXTRA IGNORED, und nur die ersten vier Zeichen würden in der Variablen abgespeichert werden. Unter ProDOS nimmt die letzte Variable einer INPUT-Liste (die Anweisung kann mehr als eine Variable enthalten) alle übriggebliebenen Zeichen in der Eingabezeile auf, eingeschlossen Kommas und Semikolons.

Nach dem Apple-Handbuch sind die Befehle IN# und PR# keine Applesoft-Befehle mehr, sondern gehören jetzt zum Betriebssystem ProDOS. Das Handbuch sagt, diese beiden Befehle würden innerhalb eines Programms nicht mehr ohne vorangestelltes Control-D funktionieren, weil sie ProDOS sonst abschalten würden. ProDOS würde sie deshalb abfangen und ignorieren. Das Handbuch hat hier Unrecht, weil die beiden Befehle IN# und PR# innerhalb eines BASIC-Programms auch ohne vorangestelltes Control-D funktionieren. Das kann verschiedene Folgen haben, die vom angewählten Steckplatz abhängen. Wenn Sie einen Steckplatz angeben, an den ein Laufwerk angeschlossen ist, versucht das System von diesem Laufwerk neu zu laden. Wenn Sie einen Steckplatz angeben, an den kein Gerät angeschlossen ist, bricht das System zusammen, und Sie müssen die Control- und die Reset-Taste drücken, um das System neu zu starten.

Die Befehle TRACE und NOTRACE hatten keine Wirkung auf DOS-Befehle. Diese Befehle, die veranlassen, daß während eines Applesoft-Programms die Zeilennummern der gerade ausgeführten Anweisungen angezeigt werden, arbeiten aber mit ProDOS- Anweisungen zusammen.

Die FRE-Anweisung gehört jetzt sowohl zu Applesoft als auch zu ProDOS. Die Applesoft-Version ist langsamer und weniger effizient als die ProDOS-Version. Mit der FRE-Anweisung wird der Speicherplatz alter und nicht mehr benötigter Zeichenketten für neue Verwendungen aufbereitet. Sie können wählen, welche Version Sie benutzen wollen; wenn Sie dem Befehl ein Control-D voranstellen, wird der schnellere ProDOS-Befehl benutzt; wenn nicht, handelt es sich um eine Applesoft-Anweisung, und der langsamere Befehl wird verwendet.

Kapitel 6 ProDOS, Speicherbelegung und periphere Geräte

In diesem Kapitel werden wir den Zusammenhang zwischen ProDOS und dem Speicher Ihres Apple II diskutieren. Das schließt eine Besprechung des von ProDOS belegten Speicherbereichs ein. Wir setzen uns mit den Befehlen auseinander, mit denen BASIC-Programmierer direkt auf Speicherstellen zugreifen können, und besprechen die Art und Weise, wie sich Applesoft und ProDOS Speicherplatz teilen. Schließlich werden wir den Zusammenhang zwischen ProDOS und peripheren Geräten behandeln, wie zum Beispiel Diskettenlaufwerken und Uhr-/Kalender-Karten.

Dieses Kapitel können Sie auf zwei verschiedene Weisen angehen. Sie können es überfliegen, wenn Sie sich nur allgemeines Hintergrundwissen über den Zusammenhang zwischen ProDOS, Speicherbelegung und peripheren Geräte verschaffen wollen. Für den mehr technisch Interessierten liefert dieses Kapitel die detaillierten Informationen, die für ein tieferes Verständnis von ProDOS notwendig sind.

ProDOS UND SPEICHERBELEGUNG

Bevor Sie ProDOS benutzen können, müssen Sie es vom Diskettenlaufwerk in den Speicher laden. Um ProDOS an die richtige Stelle im Speicher zu laden, ist eine bestimmte Abfolge von Schritten erforderlich, die Ladesequenz heißt. Wenn ProDOS arbeitet, benutzt es besondere Speicherbereiche, wie die nullte Seite (zero page) und die System-Global-Seite, um bestimmte Daten festzuhalten. Wir werden diese Daten und diese Bereiche erörtern, aber zuerst wollen wir kurz das Thema wiederholen, wie einzelne Speicherstellen im Speicher des Apple adressiert werden.

Adressierung

Der Apple kann 65536 individuelle Speicherstellen benutzen oder adressieren, in die jeweils ein Byte Information abgespeichert werden kann. Jede Stelle hat ihre eigene Adresse, ähnlich der Nummer eines Postfachs. Diese Adresse kann sowohl durch eine positive als auch durch eine negative Zahl dargestellt werden. Wenn Sie einen positiven Adressenwert haben, kann der entsprechende negative Wert berechnet werden, indem Sie davon die Zahl 65536 abziehen. Die Zahlen 100 und −65436 stellen zum Beispiel dieselbe Adresse dar. Weil 0 eine gültige Adresse ist, ist 65535 die größtmögliche gültige Adresse.

In diesem Buch werden wir Speicheradressen in dezimaler Schreibweise darstellen, wenn Sie in einer BASIC-Anweisung stehen. Sonst werden wir die hexadezimale Schreibweise benutzen. Hexadezimale Zahlen werden häufig verwendet, weil sie leichter in binäre Zahlen umgewandelt werden können als dezimale Zahlen. In Hexadezimalschreibweise wird eine Zahl mit der Basis 16 anstelle der Basis 10 dargestellt. Das bedeutet, daß einzelne Ziffern die Werte 0 bis 15 haben können. Die Buchstaben A, B, C, D, E und F sind zur Darstellung der Ziffern 10 bis 15 ausgewählt worden. Hexadezimalzahlen werden im allgemeinen durch Voranstellen des Zeichens $ als solche gekennzeichnet, damit Verwechslungen vermieden werden. Hier sind ein paar Beispiele für Hexadezimalzahlen und die ihnen entsprechenden Dezimalzahlen:

hexadezimal	**Umwandlung**	**dezimal**
$10	(1 * 16) + 0 = 16	16
$0A	10 * 1 = 10	10
$AC	(10 * 16) + (12 * 1) = 172	172
$D3	(13 * 16) + (3 * 1) = 211	211

Hexadezimalzahlen werden für gewöhnlich durch eine gerade Anzahl von Ziffern dargestellt. Das liegt daran, daß der größte Wert, der in einem einzelnen Byte gespeichert werden kann, gerade $FF ist (255 in Dezimaldarstellung). Der Wert 258 zum Beispiel muß in zwei Bytes abgespeichert werden und hat die Hexadezimaldarstellung $0102. Um zur Dezimaldarstellung zurückzukommen, multiplizieren Sie 1 * 256, und addieren Sie 2 * 1. Wie bei der dezimalen Schreibweise stellen die Ziffern auf der linken Seite der Zahl die größeren Werte dar. Weil ein solcher Wert so eng mit der Speicherkapazität eines einzelnen Bytes zusammenhängt, werden Sie im allgemeinen keine durch ein einzelnes Zeichen dargestellte Hexa-

dezimalzahl finden wie zum Beispiel $A. Statt dessen würde man $0A schreiben.

Speicherbelegung und die ProDOS-Ladesequenz

Jedesmal, wenn Sie Ihr System laden, findet eine Folge von Vorgängen statt, die Ladesequenz genannt wird. Die Aufgabe der Ladesequenz ist es, das Betriebssytem von der Diskette in den Speicher zu laden, wo es benutzt werden kann.

Ein sehr kleines einleitendes Ladeprogramm ist auf der Laufwerkkontrollkarte gespeichert. Wenn Sie das System einschalten, wird dieses Programm aktiviert. Die eigentliche Ladesequenz für ProDOS ist in den Blöcken 0 und 1 auf der Diskette gespeichert, auf die das einleitende Programm zugreift.

Die Ladesequenz sieht folgendermaßen aus:

1. Das Programm auf der Laufwerkkontrollkarte liest das Ladeprogramm auf der Diskette und legt es im Speicher ab der Adresse $0800 ab. Dann wird das Ladeprogramm ausgeführt.
2. Das Ladeprogramm sucht das ProDOS-Programm (das stets vom Typ $FF oder Systemdatei im ProDOS-Inhaltsverzeichnis ist) und lädt es an die Speicheradresse $2000. Dieses Programm enthält das Maschinensprache-Interface (MLI). Danach beginnt das Ladeprogramm mit der Ausführung des ProDOS-Programms.
3. Das MLI bestimmt die Speichergröße Ihres Computers, überprüft, welche Geräte angeschlossen sind, und bildet die Variablen und Routinen der System-Global-Seite. (Die System-Global-Seite wird in erster Linie zur Kommunikation zwischen dem Programm BASIC.SYSTEM und dem MLI verwendet.) Das MLI verlegt sich dann selbst an seinen geeigneten Speicherplatz oberhalb von $D000.
4. An dieser Stelle durchsucht das MLI das Stammverzeichnis der Ladediskette nach der ersten Datei, die die Zeichenfolge .SYSTEM in ihrem Namen enthält. (Eine solche Datei heißt auch Datei vom Typ $FF.) Das MLI lädt diese Datei an die Stelle $2000 und beginnt mit ihrer Ausführung.
5. Dann sucht das BASIC.SYSTEM nach einem Programm mit dem Namen STARTUP. Wenn es ein solches Programm findet, wird es geladen und ausgeführt. Andernfalls erscheint das BASIC-Prompt auf dem Bildschirm. Das STARTUP-Programm kann ein BASIC-, ein System-, oder ein Maschinensprache-Programm sein.

Wenn keine ProDOS-Datei oder keine Datei vom Typ $FF, welche die Zeichenfolge .SYSTEM im Namen enthält, gefunden wurde, sagt Ihnen das System, daß es nicht in der Lage ist, ProDOS zu laden. Beide Dateien müssen vorhanden sein, damit ProDOS operieren kann; sie sind die Bestandteile von ProDOS. Die STARTUP-Datei ist nur die erste Datei, die ProDOS auszuführen versucht.

Die nullte Seite (Zero Page)

Jede Seite im Speicher enthält 256 Bytes, die nullte Seite besteht also aus den Bytes $0000 bis $00FF. Applesoft und ProDOS verwenden diese Seite für bestimmte Aufgaben.

Der größte Teil der nullten Seite wird nur von Applesoft benutzt. Dieser Bereich ist aus Zeigern auf Speicherstellen zusammengesetzt, die für die Ausführung von Programmen und das Speichern von Datenfeldern gebraucht werden. Der Inhalt dieses Bereichs ändert sich ständig während der Ausführung eines Applesoft-Programms. Dort führt Applesoft Buch über solche Dinge wie, welche Zeile gerade ausgeführt wird, wo Variablenfelder beginnen und was das Ergebnis der letzten Multiplikation ist. Es gibt keinen Grund, einen dieser Werte für allgemeine BASIC-Programme zu ändern, und viele Gründe dafür, daß Sie es nicht tun sollten. Wenn Sie diese Zeiger und Kennzeichenbits (flags) unnötig verändern, werden die Ergebnisse unvorhersehbar und fast sicher bei Ihrem Programm ein Desaster anrichten. Die Verwendung der nullten Seite unter Applesoft wird im Anhang I am Ende des Buches beschrieben.

Das ProDOS-Maschinensprache-Interface (MLI) benutzt die Speicherstellen $40 bis $4E. Der Inhalt dieser Stellen wird vor ihrer Verwendung durch das MLI gesichert und nach Beendigung des MLI-Aufrufs wiederhergestellt. Sie können dort also Daten und Instruktionen speichern, ohne Gefahr zu laufen, sie bei einer MLI-Operation zu verlieren.

Die ProDOS-Laufwerkroutinen, die das MLI bei einer Diskettenoperation aufruft, belegen die Speicherstellen $3A bis $3F. Der Inhalt dieser Speicherstellen wird vor der Benutzung durch die Laufwerkroutinen nicht gerettet. Alle an den Stellen $3A bis $3F gespeicherten Daten werden durch diese Routinen ersetzt.

Die System-Global-Seite

Die System-Global-Seite ist eine spezielle Seite des Speichers, die zwischen $BF00 und $BFF (einschließlich) liegt. Dieser Bereich enthält die globalen Variablen des Systems. Die Global-Seite wird extensiv bei der

Kommunikation zwischen dem Betriebssystem und den Systemprogrammen benutzt. Neben anderen Daten enthält die System-Global-Seite die System-Bitabbildung und die Versionsnummern Ihres Systemprogramms, des MLI und des ProDOS. Der weitere Inhalt der System-Global-Seite hängt von dem Systemprogramm ab, das gerade ausgeführt wird.

Es folgt eine Besprechung der Speicherstellen, die Sie wahrscheinlich am meisten interessieren. Für eine vollständige Auflistung des Inhalts der System Global Page verweisen wir auf Anhang K in diesem Buch.

Die erste Speicherstelle der Global Page und die einzige, die Sie direkt aufrufen sollten, ist $BF00. Das ist der Eintrittspunkt für einen MLI-Aufruf. Alle MLI-Aufrufe beginnen mit einer Maschinensprache-Anweisung JSR (Jump to Subroutine) mit dieser Adresse. Die Verwendung dieser Eintrittsstelle wird in Kapitel 10 beschrieben.

Die Stelle $BF0F wird zur Ausgabe eines Fehlercodes durch das System verwendet. Wenn kein Fehler aufgetreten ist, hat dieses Byte den Wert 0.

Die Speicherstellen $BF10 bis BF2F dienen zur Führung einer Liste der Vektoren zu den Gerätetreibern. Diese Daten sind in Form von zwei Bytes langen Adressen aufgezeichnet; bei $BF10 und $BF11 steht die erste Adresse, bei $BF12 und $BF13 die zweite und so weiter. Es sind gerade soviel Bytes vorhanden, um acht Steckplätze zu versorgen, an die je zwei Laufwerke angeschlossen sind. Wenn eine dieser Adressen den Wert $D0A2 enthält, ist der entsprechende Steckplatz zur Zeit leer. ProDOS unterhält diese Liste, um zu bestimmen, wo es mit der Ausführung der Routine beginnen soll, die ein bestimmtes Laufwerk kontrolliert.

Die Bitabbildung des Speichers ist im Bereich von $BF58 bis $BF69 abgelegt. Jedes Bit eines Bytes in diesem Bereich steht für eine Seite (256 Bytes) des Speichers. Wenn der Wert eines Bits Eins beträgt, ist die entsprechende Seite belegt. Ist das Bit auf Null gesetzt, ist die Seite ungeschützt und kann zur Benutzung vergeben werden. Für eine vollständige Beschreibung der System-Bitabbildung verweisen wir auf den nächsten Abschnitt dieses Kapitels.

Indem es diese Abbildung liest, bevor es einen Puffer anlegt, verhindert ProDOS, daß Dateipuffer-Daten in geschützte Speicherbereiche geschrieben werden. Jedes Byte der System-Bitabbildung repräsentiert acht Speicherseiten. Das am weitesten links gelegene Bit in diesem Byte repräsentiert die erste Seite und das am weitesten rechts gelegene Bit die letzte Seite dieses Bereichs.

In den nächsten acht Bytes von $BF70 bis BF7F werden die Pufferadressen der offenen Dateien gespeichert. Wenn Sie diese Werte verändern, während sie aktiv sind, weiß ProDOS nicht mehr, wo der entsprechende Dateipuffer ist.

Im Speicherbereich von $BF80 bis $BF8F sind die Adressen der vier Unterbrechungsvektoren (Interrupt) von ProDOS gespeichert. (Unterbrechungen sind normalerweise von einem externen Gerät, wie etwa einem Modem, erzeugte Signale, die vom Apple eine Antwort verlangen.) Änderungen an ihren Werten während einer aktiven Unterbrechung verursachen eventuell einen Systemfehler und machen einen neuen Systemstart erforderlich, wenn die nächste Unterbrechung kommt. Sie sollten also nicht geändert werden.

In den Bytes $BF90 bis $BF92 werden Datum und Uhrzeit gespeichert. Diese Stellen werden von den Uhrzeit-/Datumsroutinen von ProDOS verwendet und können sich auch bei einer Änderung des Datums durch die Datei-Dienstprogramme auf der ProDOS User's Disk ändern. Die Diskette mit den IIc-Dienstpogrammen besitzt keine Option, mit der diese Speicherstellen beeinflußt werden können.

Byte $BF98 wird von ProDOS verwendet, um den Typ Ihres Computers festzuhalten. Eine Menge von Daten ist in dieses Byte gepackt. In den Bits 7 und 6 steht, welches Modell eines Apple II Sie benutzen. Ihre Interpretation hängt von Bit 3 ab. Wenn Bit 3 gleich 0 ist, bedeutet ein Wert 00 in den Bits 7 und 6, daß Sie einen Apple II benutzen, 01 steht für einen II+, 10 für einen IIe und 11 für einen Apple III. Ist Bit 3 gleich 1, sind die Werte 00, 01 und 11 in Bit 7 und 6 für spätere Definitionen reserviert; der Wert 10 bedeutet, daß Sie einen IIc haben.

In den Bits 5 und 4 steht, wieviel Speicherplatz ihr Computer hat. Der Wert 00 ist reserviert, der Wert 01 bedeutet 48K, der Wert 10 entspricht 64K, und 11 bedeutet 128K.

Der Wert in Bit 2 ist für zukünftige Definitionen reserviert.

Bit 1 gibt an, ob eine 80-Zeichen-Karte an Ihren Computer angeschlossen ist. Wenn ja, ist das Bit auf 1 gesetzt, andernfalls ist es 0.

Bit 0 identifiziert, ob Sie eine Uhr-/Kalender-Karte angeschlossen haben. Haben Sie eine, ist es 1, sonst ist es 0.

In Byte $BF99 ist eingetragen, welche Steckplätze belegt sind. Jedem Bit entspricht ein Steckplatz. Bit 0 repräsentiert Steckplatz 0 und Bit 7 Steckplatz 7.

Die 4 Bytes von $BFFC bis $BFFF enthalten Angaben über die Versionsnummer Ihres ProDOS und Ihrer Systemprogramme. Damit überprüfen die Systemprogramme (PRODOS und BASIC.SYSTEM) die Kompatibilität zwischen den Systemprogrammen, der MLI- und der ProDOS-Version.

Der Inhalt dieser Speicherstellen sollte unverändert bleiben. Änderungen dieser Werte können die Funktionstüchtigkeit des Betriebssystems beeinträchtigen.

DIE SYSTEM-BITABBILDUNG

Wie Sie im letzten Abschnitt erfahren haben, zeichnet die System-Bitabbildung auf, welche Speicherbereiche vom System gerade benutzt werden. ProDOS verwendet 24 Bytes (von $BF58 bis $BF69) auf der System-Global-Seite, um die Belegung der unteren 48K im RAM des Apple II festzuhalten. Ein Bit eines jeden Bytes entspricht einer einzelnen 256-Byte-Seite des RAM. Die Belegung von 1K RAM wird in 4 Bits aufgezeichnet. Wenn ein Bit den Wert 0 hat, ist die entsprechende Speicherseite frei; wenn es den Wert 1 hat, ist die Seite des RAM schon vergeben.

Die Bits innerhalb eines jeden Bytes der System-Bitabbildung werden in umgekehrter Reihenfolge verwendet. Das bedeutet, daß Bit 7 des ersten Bytes ($BF58) die ersten 256 Bytes des Speichers repräsentiert, während Bit 0 der achten Seite des Speichers entspricht. Abb. 6.1 zeigt die durch die ersten beiden Bytes der Bitabbildung repräsentierten Speicherseiten.

Die Bitabbildung repräsentiert nicht den erweiterten Speicher eines Apple IIe oder IIc und den Speicherbereich auf der Language-Karte eines Apple II+. Dieser Bereich oberhalb von 48K ist immer belegt. Das ist so,

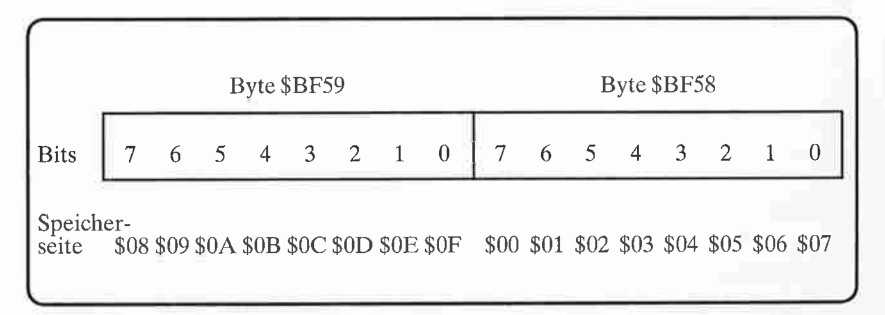


Abb. 6.1: Darstellung der Speicherbelegung durch die Bitabbildung

weil die Bitabbildung für die Verwendung durch das Maschinensprache-Interface (MLI) vorgesehen ist, das sich in diesem Speicherbereich (oberhalb von 48K) befindet.

Angenommen, Sie möchten wissen, ob eine bestimmte Seite des Speichers für die Verwendung in Ihrem Programm zur Verfügung steht. Das können Sie mit der Bitabbildung herausfinden. Die fünf oberen Bits (Bits 3–7) einer Speicheradresse liefern Ihnen die Nummer des Bytes in der Bitabbildung, das dieser Speicherseite entspricht. Das Komplement der unteren drei Bits (Bits 0–2) der Adresse gibt Ihnen das Bit innerhalb jenes Bytes an, das diese bestimmte Speicherseite repräsentiert.

Die Seite $00 wird zum Beispiel durch das erste Byte der Bitabbildung dargestellt, weil die fünf oberen Bits dieser Adresse gleich 0 sind. Die drei unteren Bits sind auch gleich 0, und das binäre Komplement zu 000 ist 111. Das ist in Dezimaldarstellung die Zahl 7, also ist die Belegung der Seite 00 in Bit 7 festgehalten.

Nur sehr fortgeschrittene Programmierer werden die System- Bitabbildung direkt benutzen. Ihre Hauptaufgabe ist es, die belegten Speicherbereiche festzuhalten und zu verhindern, daß das Betriebssystem sie noch einmal vergibt. Sie können bei der Benutzung eines bestimmten Speicherbereichs die Bitabbildung genauso verwenden, wie es das Betriebssystem macht.

Angenommen, Sie möchten sich einen Bereich des Speichers aussuchen, um dort Werte mit dem POKE-Befehl zu speichern. Indem Sie sich zuerst mit dem PEEK-Befehl die Bitabbildung anschauen und den dort gefundenen Wert analysieren, können Sie feststellen, ob eine bestimmte Speicherseite noch frei ist. Da Sie sich mit BASIC keine einzelnen Bits anschauen können, müssen Sie die ausgegebene Dezimalzahl in eine Binärzahl umformen. Wenn die Zahl größer als 127 ist, muß zum Beispiel Bit 7 gleich 1 gesetzt werden.

Da ein Mißbrauch dieser Technik zu einem Chaos im Betriebssystem führen kann, sollten Sie diese nicht erproben, ohne sich die Konsequenzen sorgfältig zu überlegen. Versuchen Sie nicht, den Inhalt der System-Bitabbildung oder der System-Global-Seite zu verändern, bevor Sie sich vergewissert haben, daß Sie die Programme im Speicher auf eine Diskette gerettet haben. Meistens ist es klüger, wenn Sie einen Weg suchen, Ihr Vorhaben ohne die Veränderung dieses Speicherbereichs durchzuführen. Beim normalen Programmieren in BASIC kann und soll die Bitabbildung ignoriert werden. Änderungen an den Werten der Bitabbildung (wie durch das „Poken“ eines neuen Wertes in ein Byte der Bitabbildung) kön-

nen bereits belegte Speicherbereiche zur neuen Benutzung freigeben, was zum Verlust von Daten und zum Stillstand des Systems führen kann.

Die System-Bitabbildung schützt keine Speicherbereiche vor dem BASIC-Befehl POKE. Sie dient nur zur Bestimmung, welche Speicherbereiche während der Ausführung eines ProDOS-Befehls belegt oder noch frei sind.

MIT BASIC AUF SPEICHERBEREICHE ZUGREIFEN

Nachdem Sie nun wissen, welchen Speicherbereich ProDOS belegt, werden Sie erfahren, wie man auf Speicherbereiche zugreift und wie man sie schützt. Im folgenden Abschnitt behandeln wir die BASIC-Anweisungen, die sich direkt auf Speicherstellen beziehen.

In Applesoft gibt es sechs BASIC-Anweisungen, mit denen man direkt auf Speicherstellen zugreifen kann. Das sind die Anweisungen PEEK, POKE, HIMEM, LOMEM, USR und CALL.

Die PEEK-Anweisung gibt den Wert der im Argument angegebenen Speicherstelle aus. Der Wert des dort gefundenen Bytes wird in dezimaler Schreibweise angegeben. Die Anweisung A = PEEK (222) würde zum Beispiel an A den in der Speicherstelle 222 gefundenen Wert übergeben; das ist die Stelle, wo der Fehlercode festgehalten wird.

Mit der POKE-Anweisung können Sie den Inhalt einer Speicherstelle ändern. In dieser Funktion sind zwei Argumente erforderlich. Das erste ist die Adresse der Speicherstelle, die Sie ändern wollen; das zweite ist der Wert, den Sie dort ablegen wollen. Der Wert des zweiten Arguments muß eine ganze Zahl zwischen 0 und 255 (einschließlich) sein. Die Anweisung POKE 9000,18 speichert zum Beispiel den Wert 18 ($12) in der Speicherstelle 9000.

Mit der Kombination der beiden Befehle PEEK und POKE können Sie jede Speicherstelle Ihres Apple II lesen und neu beschreiben. Sie können damit nach Belieben Bytes abspeichern und wieder zurückholen, Ihre Maschinensprache-Unterprogramme ablegen oder deren Ablauf durch Übergabe von Werten mit dem POKE-Befehl kontrollieren. Diese beiden Befehle bilden deshalb die Grundlage für den gesamten direkten Zugriff auf den Speicher vom BASIC aus.

Mit der CALL-Anweisung können Sie ein Maschinensprache-Unterprogramm aktivieren, das an einer von Ihnen angegebenen Stelle beginnt. Die Anfangsadresse des Unterprogramms wird als Argument der CALL-

Anweisung übergeben. Die Anweisung CALL −868 aktiviert zum Beispiel die interne Routine des Apple-Monitors, die den Text von der momentanen Cursor-Position bis zum Zeilenende löscht. Die CALL-Anweisung operiert ähnlich wie die BASIC-Anweisung GOSUB – die Ausführung Ihres Programms wird nach dem Ablauf der aufgerufenen Routine mit der nächsten Zeile hinter der CALL-Anweisung fortgesetzt. Die CALL-Anweisung kann sowohl bei schon im Speicher Ihres Apple vorhandenen als auch bei selbstgeschriebenen Routinen eingesetzt werden.

Auch mit der USR-Anweisung können Sie eine Maschinensprache-Routine aktivieren. Es gibt zwei wesentliche Unterschiede zur Operationsweise der CALL-Anweisung. Als erstes übergibt die USR-Anweisung die Kontrolle immer an eine feste Adresse, die Adresse $0A. Damit USR richtig arbeitet, müssen die Speicherstellen $0A bis $0C einen Maschinensprache-Befehl JMP mit der Anfangsadresse der auszuführenden Routine enthalten. Der zweite Unterschied besteht darin, daß die USR-Funktion als Argument einen Wert hat, der an die aufgerufene Routine übergeben wird. Nehmen wir zum Beispiel an, Sie schreiben ein Maschinensprache-Programm, das das Quadrat einer Zahl berechnet. Sie speichern dieses Programm bei der Adresse $0300. Dann legen Sie an die Adresse $0A eine JSR-Anweisung und an die Stellen $0B und $0C den Adreßwert $0300. Wenn dann in einem BASIC-Programm die Anweisung

```
PRINT USR(3) + 2
```

ausgeführt wird, erscheint die Zahl 11 auf dem Bildschirm. Diese Anweisung sagt Ihrem Computer, er soll zu dem Ergebnis der Berechnung, die das Maschinensprache-Programm an dem Argument der USR-Anweisung ausgeführt hat, die Zahl 2 addieren und das Resultat auf dem Bildschirm zeigen. Da Ihre Maschinensprache-Routine das Quadrat einer Zahl berechnet, und das Quadrat von 3 gleich 9 ist, wird bei dieser Anweisung die Zahl 11 (9+2) auf dem Bildschirm ausgegeben.

Die Bedeutung der Anweisungen CALL und USR besteht darin, daß Sie mit ihnen die Leistungsfähigkeit des ProDOS-MLI für Ihre BASIC-Programme nutzen können. Wenn Sie jedoch das MLI mehr als ein- oder zweimal in einem Programm aufrufen wollen, wird Ihnen das innerhalb eines Assembler- oder Maschinensprache-Programms viel leichter fallen. In Kapitel 10 wird das MLI besprochen.

Mit der LOMEM:-Anweisung können Sie die Adresse der unteren Grenze des für ein BASIC-Programm zur Verfügung stehenden Speicher-

bereichs festlegen. Das geschieht mit einer Anweisung wie zum Beispiel

```
LOMEM: 3055
```

die die untere Grenze des verfügbaren Speicherbereichs auf die Adresse 3055 setzt. Wenn Sie den Wert bestimmen wollen, den LOMEM zur Zeit hat, müssen Sie sich mit der PEEK-Anweisung die Stellen 106 und 105 anschauen. Dort ist dieser Wert gespeichert. Die Programmanweisung

```
A = PEEK(106) *256 + PEEK(105)
```

übergibt den Wert von LOMEM an die Variable A. Beachten Sie, daß Sie den Wert des oberen Bytes mit 256 multiplizieren müssen, um das korrekte Ergebnis zu erhalten.

Mit der HIMEM:-Anweisung können Sie die Adresse der oberen Grenze des in einem BASIC-Programm verfügbaren Speicherbereichs festlegen. Das geschieht mit einer Anweisung wie

```
HIMEM:44000
```

die HIMEM auf die Adresse 44000 im Speicher setzt. Der momentane Wert von HIMEM ist an den Stellen 116 und 115 gespeichert. Die Programmanweisung

```
A=PEEK(116)*256 + PEEK(115)
```

speichert den Wert von HIMEM in der Variablen A.

Mit den Anweisungen LOMEM und HIMEM kann man untere und obere Grenzen des für ein BASIC-Programm zur Verfügung stehenden Speicherbereichs festlegen. Diese Möglichkeit wird oft dazu benutzt, den Bereich der hochauflösenden Grafik zu schützen; das wird in Kapitel 9 im Detail besprochen.

ProDOS UND PERIPHERIEGERÄTE

ProDOS ist ein Diskettenbetriebssystem und hat natürlich hauptsächlich mit dem Betrieb von Diskettenlaufwerken zu tun. ProDOS unterstützt jedoch auch eine Uhr-/Kalender-Karte, und es bietet die Möglichkeit, bis zu vier unterbrechungsbetriebene Geräte zu bedienen. Die Zusammenarbeit von ProDOS mit solchen Geräten wird in diesem Kapitel diskutiert.

Laufwerke

ProDOS ist ein Diskettenbetriebssystem, und als solches bewahrt es Sie vor den meisten Einzelheiten eines Diskettenzugriffs. Sie brauchen nicht

unbedingt zu wissen, was für die Zusammenstellung einer Startup-Diskette notwendig ist oder in welcher Reihenfolge die Disketten nach einer bestimmten Datei durchsucht werden; das Wissen darüber verhilft Ihnen aber zu einem besseren Verständnis der Arbeitsweise von ProDOS.

Ladedisketten (Startup Disk)

Wenn Sie Ihr System laden wollen, muß sich eine gültige ProDOS-Ladediskette in einem angeschlossenen Laufwerk befinden, oder ProDOS kann den Ladevorgang nicht erfolgreich durchführen. Apple definiert eine Ladediskette als Diskette, mit der Sie ProDOS laden können. Sie muß alle Dateien enthalten, die benötigt werden, um das Betriebssystem ProDOS von der Diskette in den Speicher zu übertragen und dort mit seiner Ausführung zu beginnen. Eine solche Diskette muß folgende Eigenschaften haben:

- Sie muß für ProDOS formatiert worden sein.
- Sie muß das Programm ProDOS im Stammverzeichnis enthalten.
- Sie muß BASIC.SYSTEM oder ein anderes Programm mit der Kennung .SYSTEM enthalten. Das .SYSTEM-Programm muß vom Dateityp $FF sein und die gleichen Funktionen wie BASIC.SYSTEM ausführen.

Wenn Sie möchten, daß ProDOS sofort mit der Ausführung eines anderen Programms beginnt, muß sich im Stammverzeichnis der Diskette ein Programm mit Namen STARTUP befinden. Wenn STARTUP nicht gefunden wurde, werden Sie von ProDOS an das BASIC-Prompt übergeben.

Es ist möglich, ein Ersatzprogramm für BASIC.SYSTEM zu schreiben, solange es alle für ein solches Programm erforderlichen Konventionen und Funktionen enthält. Diese Bedingungen werden wir in Kapitel 11 erörtern.

Reihenfolge beim Absuchen der Laufwerke

Wenn Sie ProDOS auffordern, auf eine Diskette zuzugreifen, die es noch nicht kennt, muß ProDOS nach ihr suchen, um zu sehen, ob es sie finden kann. ProDOS hält dabei eine bestimmte Reihenfolge ein. Es beginnt bei der Diskette, mit der das System geladen wurde. Wenn Sie zum Beispiel das System von Steckplatz 6 geladen haben (der normalen Konfiguration), sieht ProDOS zuerst dort nach. ProDOS schaut zuerst bei Lauf-

werk 1 eines gegebenen Steckplatzes nach, bevor es bei Laufwerk 2 nachschaut.

Wenn ProDOS die Diskette mit dem gesuchten Namen nicht am Steckplatz der Ladediskette findet, durchsucht es alle Steckplätze, in denen Laufwerkkontrollkarten stecken, in absteigender Reihenfolge, angefangen bei Steckplatz 7. Angenommen Sie haben vier Laufwerke, die an Karten in den Steckplätzen 5 und 6 angeschlossen sind. Wenn die Ladediskette an Steckplatz 5 angeschlossen ist, würde ProDOS nach der neuen Diskette in folgender Reihenfolge suchen:

Steckplatz 5, Laufwerk 1
Steckplatz 5, Laufwerk 2
Steckplatz 6, Laufwerk 1
Steckplatz 6, Laufwerk 2

Wenn sich die Ladediskette im Laufwerk an Steckplatz 6 befindet und zusätzlich an Steckplatz 7 ein Festplattenlaufwerk angeschlossen wurde, ist die Reihenfolge der Suche folgendermaßen:

Steckplatz 6, Laufwerk 1
Steckplatz 6, Laufwerk 2
Steckplatz 7
Steckplatz 5, Laufwerk 1
Steckplatz 5, Laufwerk 2

Die Ladediskette ist immer die Diskette im Laufwerk mit der kleinsten Nummer, die ProDOS als Ladediskette erkennen kann. Wenn Sie zwei verschiedene Ladedisketten in den beiden Laufwerken an Steckplatz 6 haben und das System von diesem Steckplatz geladen wurde, bedeutet das, die Diskette in Laufwerk 1 wird als Ladediskette behandelt.

Uhr-/Kalender-Karten

Wenn Sie eine ThunderClock-Karte von der Firma Thunderware Inc. haben, greift ProDOS automatisch auf diese Karte zu, um Datum und Uhrzeit korrekt abzulesen. Haben Sie eine andere Uhr-/Kalender-Karte, so müssen Sie einen Gerätetreiber für diese Karte installieren. Der Hersteller einer solchen Karte kann Ihnen möglicherweise ein kurzes Programm, das als Gerätetreiber dient, mitliefern. Wenn kein Treiber erhältlich ist, können Sie ihn selbst schreiben; das können wir aber nur empfehlen, wenn Sie genügend technische Erfahrung und ein Exemplar des ProDOS Technical Reference Manual haben.

Unterbrechungsbetriebene Geräte

Einige Geräte erzeugen Unterbrechungen. Eine Unterbrechung ist ein Signal, daß der Computer beim entsprechenden Gerät nachsehen soll. Sie kommt häufig bei einem Gerät wie einem Akustikkoppler vor, der den Apple wissen lassen muß, wenn er eine Nachricht erhalten hat.

ProDOS kann bis zu vier unterbrechungsbetriebene Geräte gleichzeitig bedienen. Zu jedem solchen Gerät ist eine Unterbrechungsroutine im Speicher erforderlich. Damit ProDOS auf diese Routinen zugreifen kann, müssen Sie sie mit einem MLI-Aufruf ins System einfügen. Sie müssen auch in der System-Bitabbildung angeben, daß der von einer solchen Routine belegte Speicherbereich vergeben ist. Danach wird die Routine bei jeder Unterbrechung aufgerufen. Wenn Sie mehr als eine Unterbrechungsroutine im System installiert haben, werden diese zur Bedienung ihrer Geräte in der Reihenfolge aufgerufen, in der sie installiert worden sind. Wenn Sie den Computer abschalten, müssen diese Routinen neu installiert werden, bevor sie wieder benutzt werden können.

Wenn Sie eine Unterbrechungsroutine entfernen möchten, müssen Sie zuerst das Gerät abschalten, das sie bedient. Danach können Sie den von ihr belegten Speicherbereich in der System-Bitabbildung freigeben und die Routine mit einem MLI-Aufruf entfernen.

Eine Unterbrechungsroutine kann alles oder auch gar nichts tun. Sie muß vorhanden sein, damit beim Empfang einer Unterbrechung die Kontrolle an sie übergeben werden kann. Nach Übergabe der Kontrolle stellt ProDOS keine weiteren Anforderungen an diese Routine.

Wenn ein Akustikkoppler zum Beispiel einen Telefonanruf empfängt, erzeugt er ein Unterbrechungssignal und schickt es zum Apple. Nachdem der Computer die Unterbrechung erhalten hat, kann sie zu einer Kommunikationsroutine verzweigen, die die Arbeit ausführt, die als Antwort zu einer eingegangenen Meldung erforderlich ist.

Die MLI-Aufrufe, mit denen man Unterbrechungsroutinen installieren und entfernen kann, werden in Kapitel 10 beschrieben. Die System-Bitabbildung können Sie ändern, indem Sie mit dem POKE-Befehl den Inhalt der Speicherstelle modifizieren, die die entsprechenden Seite des Speicherbereichs repräsentiert.

Sonstige Geräte

Bisher unterstützt ProDOS keine anderen Geräte als die Laufwerke und die ThunderClock-Karte. Wenn Sie sonstige Geräte an Ihr System

anschließen wollen, müssen Sie entweder selbst die entsprechenden Gerätetreiber schreiben oder den Hersteller fragen, ob er eine solche Routine nachliefern kann. Wegen des engen Zusammenhangs eines Gerätetreibers mit der von ihm kontrollierten Hardware, werden Gerätetreiber im allgemeinen in Assembler oder Maschinensprache geschrieben und sind extrem gerätespezifisch. Deshalb können Sie im allgemeinen den Treiber vom Hersteller des Gerätes bekommen, wenn er angibt, daß sein Gerät in Ihrem System läuft.

ProDOS tut nichts, um die Benutzung weiterer Geräte zu unterstützen oder zu verhindern. Sie müssen selbst die Mittel und Wege ausarbeiten, sie zu steuern, und für die Kommunikation zwischen Ihren Programmen und den von Ihnen geschriebenen Gerätetreibern sorgen.

Wenn Sie einen IIc-Computer haben, können Sie mit Hilfe der Diskette mit den System-Dienstprogrammen die seriellen Anschlüsse auf der Rückseite Ihrer Maschine konfigurieren. Das ist ein Merkmal Ihres IIc-Computers und gehört strenggenommen nicht zu ProDOS. ProDOS ist von den Unterschieden zwischen einem IIc und einem IIe unabhängig.

Kapitel 7

Textdateien und BASIC-Programmierung unter ProDOS

In diesem Kapitel besprechen wir, wie ProDOS Textdateien handhabt; es ist für alle, die in BASIC programmieren wollen, von großem Interesse. Wir werden uns die ProDOS-Befehle zum Lesen, Schreiben und zur sonstigen Behandlung von Textdateien anschauen wie auch die BASIC-Anweisungen, die die verschiedenen Typen von Textdateien betreffen. Die Abschnitte über Programmierung von sequentiellen Dateien und Direktzugriffs-Dateien werden Ihnen zeigen, wie Sie sich die Mächtigkeit von ProDOS in Ihren BASIC-Programmen zunutze machen. Darüber hinaus können sowohl Anfänger als auch erfahrene Programmierer sehr davon profitieren, daß Sie den Gebrauch des EXEC-Befehls bei BASIC-Programmen und Textdateien lernen. Mit diesem Befehl können Sie eine ganze Reihe von Aufgaben mit einem einzigen Befehl ausführen oder eine vorherbestimmte Anzahl von Eingaben aus einer Textdatei in ein Programm machen lassen.

TEXTDATEIEN

Eine Textdatei ist eine geordnete Sammlung von Datensätzen (auch Felder genannt). Jeder Satz kann jedes beliebige Zeichen enthalten, bis auf das Return-Zeichen (ASCII 13), das das Satzenende markiert und deshalb nicht innerhalb eines Satzes verwendet werden kann. Bei ProDOS gibt es zwei verschiedene Arten von Textdateien: sequentielle Dateien und Direktzugriffs-Dateien (random access files).

Sequentielle Textdateien

Auf eine sequentielle Textdatei kann nur in einer Weise zugegriffen werden. Die Sätze in einer sequentiellen Textdatei werden in der gleichen

Reihenfolge gespeichert, wie sie in die Datei geschrieben worden sind. Der zuerst geschriebene Satz ist der erste Satz in der Datei, der zuletzt geschriebene Satz der letzte Satz in der Datei. Wenn Sie eine Datei öffnen, wird der Positionszeiger ohne Ihr Zutun auf den ersten Satz der Datei gesetzt.

Wenn Sie Sätze lesen oder in eine sequentielle Datei schreiben, können Sie sich nur vorwärts bewegen. Sie können den Zeiger niemals zurückbewegen, während die Datei geöffnet ist. Falls Sie zurückgehen müssen, sind Sie gezwungen, die Datei zu schließen, sie wieder zu öffnen und dann vorwärts, bis zum gewünschten Satz zu lesen. Das bedeutet: Sequentielle Dateien eignen sich am besten für Programme, die sie nur einmal zu lesen brauchen.

Direktzugriffs-Dateien

Eine Direktzugriffs-Datei unterscheidet sich von einer sequentiellen Textdatei in zweierlei Hinsicht: Erstens hat jeder Satz in einer Direktzugriffs-Datei eine feste Länge, während die Sätze in sequentiellen Dateien verschiedene Längen haben können. Der zweite Unterschied ist: Auf jeden Satz einer Direktzugriffs-Datei können Sie direkt zugreifen, vorausgesetzt, Sie kennen den entsprechenden Satz.

Wenn Sie eine Direktzugriffs-Datei mit der OPEN-Anweisung anlegen, müssen Sie eine Satzlänge angeben, die für jeden Satz der Datei gilt. Jedesmal, wenn Sie diese Datei wieder öffnen wollen, müssen Sie genau die gleiche Satzlänge angeben, oder Sie sind nicht in der Lage, die Datei korrekt zu lesen.

Der Hauptvorteil einer Direktzugriffs-Datei gegenüber einer sequentiellen Datei ist: Auf jeden einzelnen Satz der Datei kann direkt zugegriffen werden. Dadurch können Sie auf einen einzelnen Satz bei einer Direktzugriffs-Datei natürlich viel schneller als bei einer sequentiellen Datei zugreifen. Dies geht nur wegen der festen Satzlänge in einer Direktzugriffs-Datei.

Der Nachteil einer Direktzugriffs-Datei ist: Sie beansprucht mehr Platz auf der Diskette und erfordert etwas mehr Sorgfalt, wenn Fehler vermieden werden sollen. Deshalb unterstützen Betriebssysteme sowohl Direktzugriffs-Dateien als auch sequentielle Dateien.

Verzeichniseinträge

In einem Verzeichnis werden sowohl sequentielle als auch Direktzugriffs-Dateien als Dateien vom Typ TXT gespeichert. Den Unterschied bei bei-

den Einträgen sehen Sie nur, wenn Sie die CATALOG-Anweisung zur Ausgabe der Verzeichnisliste mit vollen 80 Spalten verwenden. Dann nämlich wird auch die Satzlänge ausgegeben. Bei Direktzugriffs-Dateien sehen Sie die Satzlänge, die Sie bei ihrer Eröffnung angegeben haben. Diese Angabe erscheint in der Form R = Satzlänge (z. B. R = 50). Bei sequentiellen Textdateien wird 0 als Satzlänge angezeigt.

Sätze

Ein Satz oder ein Feld in einer Textdatei wird immer mit einem Return-Zeichen (ASCII 13) abgeschlossen. Der erste Satz einer Datei besteht aus allen Zeichen vom Anfang der Datei bis zum ersten Return-Zeichen. Der zweite Satz enthält alle Zeichen zwischen dem ersten und dem zweiten Return-Zeichen. Wird eine PRINT-Anweisung zum Speichern von Daten in einer Datei verwendet, so wird das Return-Zeichen automatisch erzeugt. PRINT A$ zum Beispiel erzeugt nach der Ausgabe von A$ von selbst ein Return- Zeichen.

Die Apple-Handbücher verwenden in diesem Zusammenhang den Begriff Feld (field), aber Satz (record) ist ebenfalls gebräuchlich. In diesem Buch benutzen wir den Ausdruck Satz.

Datenelemente

Jeder Satz in einer Datei ist aus einem oder mehreren Datenelementen zusammengesetzt. Jedes von Applesoft unter ProDOS gespeicherte Datenelement wird vom nächsten Datenelement in diesem Satz durch ein Komma getrennt. Der Anwender ist für das Einfügen des Kommas zum Markieren des Trennpunktes zwischen zwei Datenelementen verantwortlich. Das kann mit der PRINT-Anweisung gemacht werden (wie das geht, werden wir im Abschnitt über BASIC- Anweisungen behandeln).

Die Teile eines Satzes werden oft Felder genannt. Da aber Sätze ebenfalls oft als Felder bezeichnet werden, würde ein solcher Ausdruck nur Verwirrung stiften. Wir werden ihn daher nicht verwenden und benutzen statt dessen wie das Apple-Handbuch den Ausdruck Datenelemente (data elements) im Verlauf dieses Buches.

Positionszeiger

Für jede offene Datei unterhält ProDOS einen Zeiger, der Ihre momentane Position innerhalb der Datei anzeigt. Wenn Sie aus einer Datei gelesen oder auf eine Datei geschrieben haben, weist dieser Zeiger hinter das Zeichen, das zuletzt gelesen oder geschrieben worden ist. Der Wert die-

ses Zeigers kann mit den Parametern F und B in den Anweisungen READ, WRITE und POSITION geändert werden.

Es gibt keinen BASIC- oder ProDOS-Befehl, der den momentanen Wert des Positionszeigers angibt. Diesen Wert können Sie durch Zählen eines jeden gelesenen oder geschriebenen Bytes in einem Programm bestimmen.

Die Parameter L , F und R

Wir haben schon vorher erwähnt, daß Sie beim Anlegen einer Direktzugriffs-Datei eine Satzlänge angeben müssen, die dann für jeden Satz der Datei gilt. Das geschieht mit dem Parameter L in der OPEN-Anweisung. Wenn Sie die Datei erneut öffnen, muß der Parameter L den gleichen Wert haben, den Sie ihm beim Anlegen der Datei gegebenen haben. Sonst können Sie die Datei nicht richtig lesen.

Nehmen wir zum Beispiel an, Sie haben eine Datei namens TESTDATEI mit der Satzlänge 50 auf der Diskette. Falls Sie versuchen, diese Datei mit der Anweisung

```
OPEN TESTDATEI,L50
```

zu öffnen, geraten Ihre Daten hoffnungslos durcheinander. Werden Daten in eine Direktzugriffs-Datei geschrieben, erhalten alle Sätze die Satzlänge, die Sie in der OPEN-Anweisung angegebenen haben. Falls Sie zum Lesen eine andere Satzlänge angeben, berichtigt ProDOS die Länge nicht. Sie erhalten verstümmelte Sätze, weil ProDOS dann zum Beispiel versucht, zehn 30 Bytes lange Sätze aus einer Datei zu lesen, die zehn Sätze der Länge 50 enthält.

Wie wir im Kapitel über den Positionszeiger erwähnt haben, können Sie mit den Parametern F und B bei einigen ProDOS-Befehlen den Positionszeiger einer Datei vorwärtsstellen. Sie können damit jedoch nicht den Zeigerwert verringern oder in einer Datei zurückgehen.

Der Parameter F rückt die Zeigerposition um ganze Sätze vor. Das bedeutet, eine Anweisung wie

```
READ MEINEDATEI,F5
```

würde den Inhalt von fünf vollständigen Sätzen lesen und ignorieren, bevor sie den Inhalt des sechsten Satzes an das Programm übergibt. ProDOS erkennt ein Satzende am Return-Zeichen (ASCII 13).

Der Parameter B rückt die Zeigerposition um einzelne Bytes vor. Das heißt, eine Anweisung wie

```
READ MEINEDATEI,B10
```

würde zehn Bytes Daten auf der Diskette lesen und ignorieren, bevor sie den Inhalt des nächsten Bytes an Ihr Programm übergibt. Dabei können ein oder mehrere kurze Sätze überlesen werden. Sie können den Parameter B dazu verwenden, eine bestimmte Zeichenfolge in einer Datei zu suchen oder zu erzeugen.

Die Parameter F und B können zusammen benutzt werden. In diesem Fall wird der Parameter F zuerst ausgeführt. Die Anweisung

```
READ MEINEDATEI,F5,B10
```

veranlaßt ProDOS, zuerst fünf vollständige Sätze und dann 10 einzelne Bytes von MEINEDATEI zu überlesen, bevor es einen Satz an Ihr Programm übergibt.

Bei Dateien für den wahlfreien Zugriff können Sie mit dem Parameter R die momentane Zeigerposition auf einen bestimmten Satz setzen. Die Anweisung

```
READ MEINEDATEI,R10
```

würde Satz 10 in MEINEDATEI lesen. ProDOS berechnet die Position eines Satzes, indem es die Nummer des gewünschten Satzes (10 in diesem Fall) mit der Satzlänge multipliziert.

Obwohl die Parameter F und R das gleiche bewirken, ist die Verwendung des Parameters R (der nur bei Direktzugriffs-Dateien benutzt werden kann) zum Zugriff auf einen bestimmten Satz viel schneller. Das liegt daran, daß ProDOS nicht alle Sätze von der momentanen Zeigerposition bis zum gewünschten Satz zu lesen (und zu ignorieren) braucht. Der zweite Vorteil des Parameters R ist: Man kann sich innerhalb einer Datei vorwärts und rückwärts bewegen.

ProDOS-ANWEISUNGEN

Die Anweisungen OPEN, CLOSE, WRITE, FLUSH, READ, POSITION und APPEND sind ProDOS-Befehle. Wenn man sie von einem BASIC-Programm aus benutzen will, müssen sie als Zeichenkette in eine PRINT-Anweisung eingebettet werden. Vor diese Zeichenkette muß noch das Control-D-Zeichen gesetzt werden, damit ihr Inhalt als ProDOS-Befehl interpretiert werden kann. Dieses Zeichen kann mit der

Anweisung CHR$(4) erzeugt werden. Es wird so häufig gebraucht, daß man es oft am Anfang eines Programms einer Variable zuweist (i. a. D$). Schauen Sie sich als Beispiel das folgende kurze Programm an:

```
10 D$ = CHR(4)
20 PRINT D$; "OPEN BEISPIEL"
30 PRINT D$; "WRITE BEISPIEL"
40 PRINT "TESTDATEN"
50 PRINT D$; "CLOSE BEISPIEL"
```

ProDOS erkennt, daß die Anweisungen in den Zeilen 20, 30 und 50 für es selbst bestimmt sind; die PRINT-Anweisung in Zeile 40 wird ProDOS jedoch ignorieren. Der Grund dafür ist: Die in den Zeilen 20, 30 und 50 gedruckten Zeichenketten beginnen mit dem Zeichen Control-D (das durch den Wert der Variablen D$ dargestellt wird).

Die Befehle CLOSE und FLUSH können im Direktmodus von BASIC angewendet werden. Tippen Sie dazu einfach den Befehl ein, und drükken Sie die Return-Taste. Die anderen Befehle gibt es nicht im Direktmodus.

Die OPEN-Anweisung

Bevor Sie irgend einen anderen Befehl zur Bearbeitung einer Datei verwenden können, muß die OPEN-Anweisung erfolgt sein. Die OPEN-Anweisung muß als Argument den Pfadnamen der Datei, die Sie öffnen wollen, enthalten. Bei einer Direktzugriffs-Datei müssen Sie auch die Satzlänge angeben. Zusätzlich können sie Steckplatz- und Laufwerknummer angeben.

Wenn Sie identische Disketten in Ihren Laufwerken haben und nicht wollen, daß ProDOS die Datei bearbeitet, die es nach der Reihenfolge, in der es sucht, zuerst findet, können Sie die Nummern von Steckplatz und Laufwerk angeben, in dem sich die Diskette befindet. Die Anweisung

```
OPEN MEINEDATEI,S5,D1
```

würde zum Beispiel sicherstellen, daß ProDOS die Diskette in Laufwerk 1 an Steckplatz 5 verwendet. Sie können die Parammeter S und D zusammen und einzeln benutzen.

Die OPEN-Anweisung weist jeder Datei, die Sie mit dieser Anweisung in einem Programm öffnen, einen Puffer von 1024 Bytes im Speicher zu. Wenn Sie eine Datei nicht geöffnet haben, sind Sie nicht in der Lage, von ihr zu lesen oder auf sie zu schreiben, weil kein Puffer zum Ablegen der Daten existiert.

Die CLOSE-Anweisung

Die CLOSE-Anweisung zum Schließen einer Datei muß von Ihrem Programm ausgegeben werden, um sicherzustellen, daß alle Daten im Dateipuffer zur Diskette geschrieben werden und daß der Pufferbereich wieder freigegeben wird. Vorher kann der Speicherbereich des Puffers nicht neu verwendet werden. Wenn nach Ende Ihres Programms nicht alle Dateien geschlossen worden sind, erhalten Sie die Fehlermeldung

```
FILES STILL OPEN
```

Tippen Sie in diesem Fall CLOSE ein, und drücken Sie die Return-Taste, um alle offenen Dateien zu schließen.

Wenn Sie die Anweisung CLOSE alleine verwenden, werden alle offenen Dateien geschlossen. Wollen Sie eine spezielle Datei schließen, müssen Sie den Pfadnamen dieser Datei in der CLOSE- Anweisung angeben. Die Anweisung CLOSE MEINEDATEI am Ende Ihres Programms würde nur diese Datei schließen und alle anderen offenen Dateien unberührt lassen.

Sequentielle Dateien können nur in einer Richtung gelesen werden. Aus diesem Grunde müssen Sie, wenn Sie den zweiten Satz in einer solchen Datei lesen wollen und sich gerade beim dritten Satz befinden, die Datei schließen und neu öffnen, um an den Anfang zurückzukommen. In diesem Fall würden Sie nur diese eine Datei schließen wollen, ohne die anderen Dateien zu beeinflussen. Wenn Sie die Datei neu geöffnet haben, können Sie sie vorwärts lesen bis zu dem Satz, den Sie sehen möchten.

Die WRITE-Anweisung

Bevor Sie die PRINT-Anweisung zum Speichern von Daten in einer Datei benutzen können, müssen Sie die WRITE-Anweisung eingeben. Die Daten werden immer von der momentanen Zeigerposition an gespeichert. Mit dem Parameter F können Sie bei dieser Anweisung die Zeigerposition vorrücken lassen. Der Wert dieses Parameters gibt die Anzahl der Sätze an, die ProDOS überlesen soll. ProDOS zählt dazu die Return-Zeichen (ASCII 13) hinter der gegenwärtigen Zeigerposition und vergleicht sie mit dem Wert des Parameters R. Sie können die Zeigerposition innerhalb einer Datei auch mit dem Parameter B vorrücken lassen. Hier gibt der Parameter die Anzahl der Bytes an, die überlesen werden sollen.

Die Anweisung

```
WRITE BEISPIEL,F5,B5
```

zum Beispiel veranlaßt ProDOS, fünf Sätze von der momentanen Position in der Datei aus zu überlesen und dort den Zeiger neu zu plazieren. Danach überliest ProDOS weitere fünf Bytes und bringt den Zeiger an diese Stelle. Von hier aus beginnt ProDOS mit dem Schreiben von Daten, wenn es in einer nachfolgenden PRINT-Anweisung dazu aufgefordert wird. Ein WRITE-Befehl bleibt bis zur Ausführung eines anderen ProDOS-Befehls in Kraft.

Mit dem Parameter R in dieser Anweisung können Sie bei einer Direktzugriffs-Datei einen bestimmten Satz angeben. Das geht nur bei Direktzugriffs-Dateien, weil dort alle Sätze die gleiche Länge haben. Die Anweisung

```
WRITE DIREKT.DATEN,R50
```

würde ProDOS zum Beispiel veranlassen, die momentane Zeigerposition an den Anfang von Satz 50 zu verschieben. ProDOS macht das, indem es die Anzahl der zu überspringenden Sätze (vom Anfang der Datei) mit der Satzlänge multipliziert. Mit diesem Parameter können Sie auch eine Stelle vor der momentanen Zeigerposition erreichen.

Die FLUSH-Anweisung

Die FLUSH-Anweisung sorgt dafür, daß alle Daten in den Dateipuffern auf die Diskette übertragen werden. Wenn Sie bei der FLUSH-Anweisung den Pfadnamen einer Datei angeben, wird nur der Puffer dieser Datei auf die Diskette geschrieben. FLUSH löst den Dateipuffer nicht auf, wie das die CLOSE-Anweisung macht, und Sie brauchen die Datei nicht neu zu öffnen, wenn Sie wieder lesen oder in die Datei schreiben wollen. Wie schon vorher in diesem Buch erwähnt, beruht die größere Geschwindigkeit von ProDOS zum großen Teil darauf, daß ProDOS wartet, bis ein Dateipuffer voll ist, bevor es ihn auf die Diskette schreibt. Eine häufige Verwendung des FLUSH-Befehls innerhalb eines Programm verhindert den Verlust der Daten in den Dateipuffern durch einen Unfall, verlangsamt aber andererseits die Ausführung eines Programms.

Die READ-Anweisung

Bevor Sie Daten aus einer Datei zurückerhalten können, muß die READ-Anweisung eingegeben werden. Vor der READ-Anweisung muß die Datei geöffnet werden, oder Sie begehen einen Fehler. In diesem Fall erscheint die Meldung

```
FILE NOT OPEN
```

Sie können die Parameter F, B und R hier genau wie bei der WRITE-Anweisung benutzen.

Wie bei der WRITE-Anweisung bleibt ein READ-Befehl bis zur Eingabe des nächsten ProDOS-Befehls gültig. Wenn Sie also eine Datei auf READ gesetzt haben, holt Ihr Programm die von allen INPUT- und GET-Anweisungen angeforderten Daten aus dieser Datei. Sie können während der Gültigkeit dieses Befehls keine Daten über die Tastatur eingeben. Wenn eine Eingabe über die Tastatur nötig ist, können Sie mit einem ProDOS-Befehl wie FRE die Wirkung der READ-Anweisung aufheben.

Die POSITION-Anweisung

Auch die POSITION-Anweisung kann zur Änderung der momentanen Zeigerposition innerhalb einer Datei verwendet werden, und das ist schon alles, was Sie mit dieser Anweisung machen können. Sie wurde bei ProDOS aus Gründen der Kontinuität mit DOS 3.3 aufgenommen; die Möglichkeit, bei den Aweisungen READ und WRITE die Parameter F und B zu benutzen, machen die POSITION-Anweisung überflüssig.

Die POSITION-Anweisung hat die gleiche Form wie die Anweisungen READ und WRITE. Die Anweisung

```
POSITION BEISPIEL,F5,B5
```

veranlaßt ProDOS also, fünf Sätze zu überspringen und den Positionszeiger dementsprechend zu berichtigen; dann überspringt ProDOS weitere fünf Bytes und bringt den Positionszeiger wieder auf den neuen Stand.

Die APPEND-Anweisung

Mit der APPEND-Anweisung können Sie Sätze ans Ende einer Datei schreiben. APPEND wirkt als Abkürzung für drei andere Befehle: OPEN, POSITION und WRITE. Wenn Sie unter ProDOS einen APPEND-Befehl eingeben, wird die Datei geöffnet; das Ende der Datei wird zur neuen Zeigerposition. Dabei verhält sich das System so, als ob ein WRITE-Befehl eingegeben worden wäre. Mit dem APPEND- Befehl sparen Sie auch Programmzeilen und Ausführungszeit in Ihren Programmen. Sie brauchen keine Anweisungen zum Durchlesen einer Diskettendatei und zum Auffinden des Dateiendes zu schreiben. Wenn Sie den APPEND-Befehl bei einer Direktzugriffs-Datei verwenden, müssen Sie mit dem Parameter L eine Satzlänge angeben. Sie können bei diesem Befehl auch die Parameter für Steckplatz und Laufwerk verwenden.

BASIC-ANWEISUNGEN

Die folgenden Anweisungen sind normale Applesoft-Befehle. Wenn Sie diese jedoch in einem Programm hinter einer ProDOS-Anweisung READ oder WRITE benutzen, ändern sie ihr Verhalten. Nach Aufhebung der Anweisungen READ oder WRITE durch einen anderen ProDOS-Befehl verhalten sie sich wieder wie vorher.

Die PRINT-Anweisung

Solange ein WRITE-Befehl gültig ist, überträgt die PRINT-Anweisung Daten in die im letzten WRITE-Befehl angegebene Diskettendatei. Das sind die Daten, die hinter der PRINT-Anweisung spezifiziert wurden, Variablen, in Anführungszeichen stehende Zeichenketten oder numerische Werte. Alle PRINT-Anweisungen, die eingegeben werden, während der WRITE-Befehl gültig ist, übertragen ihre Daten auf die Diskettendatei und nicht auf den Bildschirm. Sobald ein anderer ProDOS-Befehl ausgeführt wurde, überträgt eine PRINT-Anweisungen keine Daten mehr auf die Diskette und hat wieder ihre normale Funktion.

Die PRINT-Anweisung druckt normalerweise hinter die Daten ein Return-Zeichen (ASCII 13). Mit diesem Zeichen wird bei ProDOS das Ende eines Satzes markiert. Sie können eine solche Markierung des Satzendes verhindern, indem Sie Ihre PRINT-Anweisung mit einem Semikolon beenden. Wenn die PRINT-Anweisung auf dieses Zeichen stößt, überträgt sie kein Return-Zeichen auf die Diskette. Das ist wichtig, weil Sie so mit mehreren PRINT-Anweisungen Daten in den gleichen Satz einer Diskettendatei schreiben können.

Nehmen wir zum Beispiel an, daß die Variable A$ die Zeichenkette KLAVIER und die Variable B$ die Zeichenkette GITARRE enthält. Die folgenden Beispiele zeigen Ihnen die Wirkung des Semikolons auf die Ausgabe der PRINT-Anweisung:

Anweisung	**Ausgabe**
PRINT A$	KLAVIER<CR>
PRINT A$;	KLAVIER
PRINT A$;B$;	KLAVIERGITARRE
PRINT A$;"GEIGE"	KLAVIERGEIGE<CR>

Oben ist die Ausgabe einzelner Anweisungen zu sehen. Die Wirkung bleibt von einer Anweisung zur nächsten erhalten. Zum Beispiel:

Anweisung	**Ausgabe**
PRINT A$; PRINT B$	KLAVIERGITARRE<CR>

Sie stellen fest, daß die letzten beiden Anweisungen einen einzigen Satz bilden, der mit einem Return-Zeichen (hier durch das Symbol <CR> dargestellt) endet. Das Semikolon am Ende der ersten Anweisung verhindert das Return-Zeichen und somit das Satzende. Die zweite Anweisung hört nicht mit einem Semikolon auf, also wird ein Return-Zeichen erzeugt und damit das Ende des Satzes markiert. Wenn der ProDOS-Befehl WRITE in Kraft ist, wird die Ausgabe auf eine Diskettendatei übertragen.

Das Komma kann bei der PRINT-Anweisung als Editierzeichen verwendet werden. Wenn Sie ein Komma als Fortsetzungsmarke in einer PRINT-Anweisung benutzen, werden keine Leerzeichen zwischen den zur Diskette übertragenen Zeichen eingefügt. Die Daten vor und hinter dem Komma werden so zusammengefügt, als ob Sie eine Verkettung mit dem +-Operator vorgenommen hätten. Die zusammengefügten Daten werden in den gleichen Satz der Datei geschrieben.

Nehmen wir wieder an, die Variablen A$ und B$ enthalten die Zeichenketten KLAVIER und GITARRE. Die Wirkung des Kommas bei einer PRINT-Anweisung sieht folgendermaßen aus:

Anweisung	**Ausgabe**
PRINT A$,"TEST"	KLAVIERTEST<CR>
PRINT A$,B$	KLAVIERGITARRE<CR>
PRINT A$,	KLAVIER<CR>
PRINT A$,",",B$	KLAVIER,GITARRE<CR>
PRINT A$,B$;	KLAVIERGITARRE

Beachten Sie: Sie können ein Semikolon innerhalb der gleichen Anweisung anstelle eines Kommas nehmen, und ein Komma am Ende einer Anweisung unterdrückt nicht das Return-Zeichen. Im vierten Beispiel wird ein Komma tatsächlich auf die Diskette ausgegeben, weil es in einer Zeichenkette enthalten ist. Der Einschluß eines Kommas in eine Zeichenkette oder eine Variable ist die einzige Möglichkeit, mit der PRINT-Anweisung Datenelemente innerhalb eines Satzes zu erzeugen.

Die INPUT-Anweisung

Die INPUT-Anweisung holt Daten von der Diskette anstatt von der Tastatur, wenn sie in einem Programm hinter einer READ-Anweisung

benutzt wird. Bei der INPUT-Anweisung können Sie das Komma dazu verwenden, zwei oder mehr Datenelemente eines Satzes zu lesen. Wenn die INPUT-Anweisung zwei durch ein Komma getrennte Variablen enthält, liest ProDOS zwei Datenelemente eines Satzes. Enthält der Satz mehr als zwei Datenelemente, wird der Rest des Satzes nicht beachtet. Wenn der Satz nur ein Datenelement enthält, wird das zweite Element automatisch dem nächsten Satz entnommen.

Die GET-Anweisung

Die GET-Anweisung holt ein Zeichen von der Diskette statt von der Tastatur, wenn es hinter einer ProDOS-Anweisung READ verwendet wird. Die GET-Anweisung unterscheidet sich von der READ-Anweisung dadurch, daß sie immer nur ein Zeichen liest und daß sie jedes Zeichen liest. Das bedeutet, die GET-Anweisung liest auch ein Komma oder ein Return-Zeichen ein, wenn sie eins findet. Sie müssen bei der Verwendung dieses Befehls jedes Byte korrekt interpretieren. Die GET-Anweisung kann wertvoll zum Lesen von Dateien sein, die entweder eine unbekannte Anzahl von Elementen pro Satz oder eine variable Anzahl von Elementen pro Satz oder Text besitzen, der Satzzeichen enthält.

Bei der Verwendung der GET-Anweisung zum Lesen von Daten aus einer Textdatei merkt ProDOS nicht, wann das Ende eines Satzes erreicht ist. Das liegt daran, daß jedesmal nur ein Zeichen eingelesen wird. Deshalb müssen Sie, wenn Sie sich die Daten in einer Textdatei als eine Reihe von Sätzen anschauen wollen, überprüfen, ob das eingelesene Zeichen ein Return-Zeichen (ASCII 13) ist. Ihr Programm müßte einen Test folgender Art enthalten:

```
1000 GET A$
1010 IF A$ = ASC(13) THEN GOSUB 2000
```

Bei diesem Beispiel fängt in der Zeile 2000 ein Unterprogramm an; es führt aus, was geschehen soll, sobald das Ende eines Satzes erreicht ist. Das gleiche würde für jedes andere Zeichen gelten, das Sie aufspüren möchten. Das Programm muß jedes von der Datei eingelesene Zeichen individuell überprüfen, um festzustellen, ob es sich um das gesuchte Zeichen handelt, und eine auf diesem Ergebnis beruhende Entscheidung treffen.

Der Vorteil der GET-Anweisung beim Lesen von Daten einer Diskette besteht darin, daß Sie jedes Zeichen auf der Diskette lesen können und Ihre Entscheidungen von der Bedeutung eines jeden einzelnen Zeichens abhängig machen können.

PROGRAMMIEREN MIT SEQUENTIELLEN DATEIEN

Eine sequentielle Datei zeichnet sich durch ein bestimmtes Merkmal aus: Sie kann vom Anfang bis zum Ende nur in eine Richtung gelesen werden. In diesem Kapitel zeigen wir Ihnen, wie Sie unter ProDOS sequentielle Dateien anlegen, lesen und schreiben können.

Eine sequentielle Datei anlegen

Das Anlegen einer sequentiellen Datei ist sehr einfach; es erfordert nur eine OPEN-Anweisung und den Pfadnamen der Datei, die Sie anlegen wollen. Wenn Sie eine OPEN-Anweisung an ProDOS ausgeben, legt es eine neue Datei auf der Diskette an, falls nicht schon eine mit demselben Pfadnamen existiert. Die neue Datei besteht aus einem Verzeichniseintrag und einer leeren nicht indizierten Datei aus 512 Bytes. Versuchen Sie, aus einer leeren Datei Daten zu lesen, erscheint die Fehlermeldung END OF DATA. Wenn eine Datei angelegt und geöffnet worden ist, müssen Sie zuerst mit der PRINT-Anweisung Daten in die Datei schreiben, bevor Sie irgendwelche Daten aus ihr lesen können. Schreiben Sie mehr Daten auf die Datei, so wird ProDOS die Datei auf eine einfach indizierte und dann auf eine zweifach indizierte Datei erweitern, falls die entsprechende Dateigröße erreicht ist.

Das folgende Programm legt eine sequentielle Datei an und schreibt eine Reihe von Sätzen hinein. Wenn Sie dieses und die anderen Programme in diesem Abschnitt eingeben und testen, sollten Sie sie mit dem SAVE-Befehl auf einer Diskette abspeichern. Die Programme in diesem Abschnitt hängen alle miteinander zusammen, und Sie möchten Sie vielleicht später noch einmal aufrufen.

```
  10 REM LEGE SEQUENTIELLE DATEI AN
  20 D$ = CHR$(4)
 100 HOME
 110 VTAB 2: PRINT "LEGE DATEI AN"
 120 GOSUB 1000
 199 END
1000 PRINT D$;"OPEN SEQ.DATEN"
1010 PRINT D$;"WRITE SEQ.DATEN"
1020 PRINT "PULT"
1021 PRINT "STIFT"
1022 PRINT "PAPIER"
1025 PRINT "STUHL"
1026 PRINT "TISCH"
```

```
1027 PRINT "SCHEMEL"
1030 PRINT "LAMPE"
1031 PRINT "BUECHER"
1032 PRINT "BUECHERREGAL"
1040 PRINT D$;"CLOSE SEQ.DATEN"
1099 RETURN
```

Das obige Programm schreibt eine Reihe von Zeichenketten in die Diskettendatei. Es gibt keinen besonderen Grund, gerade die Zeichenketten in den Zeilen 1020 bis 1032 zu verwenden. Wenn Sie Ihre eigenen Daten eingeben wollen, können Sie dieses Programm modifizieren, indem Sie die Zeilen 1020 bis 1032 löschen und sie durch folgende ersetzen:

```
1020 INPUT A$
1022 IF A$="ENDE" THEN GOTO 1040
1024 PRINT A$
1026 GOTO 1020
```

Nach dieser Änderung können Sie Ihre eigenen Daten nach Belieben eingeben. Wenn Sie die Zeichenkette ENDE als Anwort auf Zeile 1020 eingeben, wird A$ nicht mehr auf die Diskette geschrieben, und die Datei wird geschlossen, bevor das Programm endet.

Eine sequentielle Datei lesen

Nachdem Sie mit dem vorherigen Programm eine sequentielle Datei angelegt haben, müssen Sie wissen, wie Sie die Daten von der Datei wieder einlesen können. Das folgende kurze Programm öffnet die Datei SEQ.DATEN, liest sie und gibt die Daten auf dem Bildschirm aus. Falls Sie versuchen, dieses Programm laufen zu lassen, ohne vorher eine Datei mit dem Namen SEQ.DATEN angelegt zu haben, erhalten Sie die Fehlermeldung END OF DATA, wenn Sie versuchen, die READ-Anweisung zu benutzen. Das liegt daran, daß die OPEN-Anweisung in diesem Programm die Datei anlegt, wenn sie sie nicht findet, und die Datei ist dann leer.

```
  10 REM SEQUENTIELLE DATEI LESEN
  20 D$ = CHR$(4)
  30 A = 1
 100 HOME
 110 VTAB 2: PRINT "LESE DATEI"
1000 ONERR GOTO 1040
1005 PRINT D$;"OPEN SEQ.DATEN"
```

```
1010 PRINT D$;"READ SEQ.DATEN"
1020 INPUT B$(A)
1025 A = A + 1
1030 GOTO 1020
1040 PRINT D$;"CLOSE SEQ.DATEN"
1050 POKE 216,0
2000 FOR A = 1 TO 9
2010 PRINT "SATZ";A;" = ";B$(A)
2020 NEXT A
```

Dieses Programm verwendet eine bekannte Prozedur zum Lesen von Sätzen aus einer sequentiellen Datei. Das Programm durchläuft die Schleife von 1020 bis 1030 bis die ONERR-Anweisung in Zeile 1000 bei Erreichen des Dateiendes aktiviert wird. In diesem Fall verzweigt das Programm nach Zeile 1040 und schließt die Datei. Die POKE- Anweisung in Zeile 1050 setzt das Fehler-Kennzeichenbit wieder auf Null zurück. Das Bit ist gesetzt worden, als das Programm versuchte, mehr Sätze aus der Datei zu lesen, als darin vorhanden waren. Das ist eine sehr bequeme Technik. Sie ist besonders nützlich, wenn Sie nicht wissen, wieviel Sätze die Datei enthält.

Nachdem die Datei geschlossen wurde, wird in der FOR/NEXT-Schleife in den Zeilen 2000 bis 2020 der Inhalt der Sätze auf dem Bildschirm ausgegeben, bevor das Programm endet.

An eine sequentielle Datei anfügen

Wie wir bereits vorher in diesem Kapitel erwähnt haben, können Sie den APPEND-Befehl dazu verwenden, eine Datei zu öffnen, die gegenwärtige Zeigerposition ans Ende der Datei zu verlegen und eine WRITE-Anweisung auszugeben. Das folgende Programm ist ein Beispiel für die Verwendung der APPEND-Anweisung bei einer sequentiellen Datei:

```
  10 REM AN EINE SEQUENTIELLE DATEI ANFUEGEN
  20 D$ = CHR(4)
 100 HOME
 110 VTAB 2: PRINT "FUEGE AN DATEI AN"
 120 GOSUB 1000
 199 END
1000 PRINT D$;"APPEND SEQ.DATEN"
1020 PRINT "SCHACHTEL"
1021 PRINT "ETUI"
1022 PRINT "TASCHE"
```

```
1025 PRINT "FERNSEHER"
1026 PRINT "ZIMMER"
1027 PRINT "BILD"
1030 PRINT "KARTE"
1031 PRINT "MONITOR"
1032 PRINT "COMPUTER"
1040 PRINT D$;"CLOSE SEQ.DATEN"
1099 RETURN
```

Wie das frühere Programm zum Anlegen einer sequentiellen Datei schreibt auch dieses Programm nur eine Anzahl von Zeichenketten auf eine Diskettendatei. Es gibt keinen besonderen Grund, die Zeichenketten in den Zeilen 1020 bis 1032 zu verwenden. Wenn Sie Ihre eigenen Daten eingeben möchten, können Sie dieses Programm wieder modifizieren, indem Sie die Zeilen 1020 bis 1032 löschen und sie durch folgende ersetzen:

```
1020 INPUT A$
1022 IF A$="ENDE" THEN GOTO 1040
1024 PRINT A$
1026 GOTO 1020
```

Nach dieser Änderung können Sie beliebige eigene Daten eingeben. Wenn Sie die Zeichenkette ENDE als Antwort auf Zeile 1020 eingeben, wird A$ nicht auf die Diskette geschrieben, und das Programm endet nach dem Schließen der Datei. Da dieses Programm den APPEND-Befehl als Ersatz für die Befehle OPEN und WRITE verwendet, werden alle Daten, die Sie eingeben, ans Ende der Datei angefügt.

PROGRAMMIEREN MIT DIREKTZUGRIFFS-DATEIEN

Direktzugriffs-Dateien haben die folgenden beiden charakteristischen Eigenschaften: Erstens haben alle Sätze in der Datei die gleiche Länge, zweitens kann jeder einzelne Satz in der Datei zu jeder Zeit gelesen werden. Das bedeutet, Sie können sich zum Lesen oder Schreiben eines Satzes innerhalb einer Datei sowohl vorwärts als auch rückwärts bewegen. Der folgende Abschnitt zeigt, wie man die ProDOS-Dateien für den direkten Zugriff von BASIC aus benützt.

Eine Direktzugriffs-Datei anlegen

Wie wir oben erwähnt haben, gibt der Parameter L in der OPEN-Anweisung ProDOS beim Anlegen einer Direktzugriffs-Datei die Länge eines

jeden Satzes an. Diese Länge wird durch eine Anzahl von Bytes ausgedrückt. Sie muß groß genug sein, um alle Daten im größten Satz der Datei enthalten zu können, zuzüglich eines Bytes für das Return-Zeichen.

Wenn Sie eine WRITE-Anweisung zum Abspeichern von mehr Daten verwenden, als in einen einzelnen Satz hineinpassen, werden die überzähligen Daten in den nächsten Satz geschrieben. Das kann katastrophale Folgen haben. Die im folgenden Satz bereits vorhandenen Daten können unlesbar werden. Wenn Sie nicht sicher sind, daß die Satzlänge groß genug ist, empfiehlt es sich im allgemeinen, einen Test in Ihre Programme einzufügen, der einen Überlauf verhindert.

Wenn Sie zum Beispiel eine Datei mit der Satzlänge 10 geöffnet haben und die Zeichenkette DAS IST EIN TEST in den ersten Satz schreiben wollen, werden die letzten sechs Zeichen in den zweiten Satz geschrieben. Der erste Satz würde DAS IST EI und der zweite Satz N TEST enthalten zuzüglich des alten Inhalts der letzten vier Bytes. Wenn Sie das verhindern wollen, können Sie die Länge der Zeichenkette, die Sie auf Diskette schreiben wollen, mit der Satzlänge der Datei vergleichen. Ist die Zeichenkette zu lang, könnte Ihr Programm eine Neueingabe der Daten erlauben. Eine weitere Methode zur Behandlung dieses Problems wäre das Abschneiden aller überlaufenden Daten. Damit würden Sie unter Umständen wichtige Daten am Ende einer Zeichenkette verlieren, aber die Zerstörung des folgenden Satzes verhindern.

Wenn Sie Daten in eine Direktzugriffs-Datei schreiben, müssen Sie die Nummer des zu verwendenden Satzes angeben, oder ProDOS schreibt die Daten automatisch in Satz 0. Dabei gehen die alten Daten in Satz 0 verloren. Benutzen Sie also den Parameter R bei der WRITE-Anweisung, um die Satznummer anzugeben.

Das folgende Programm legt eine Direktzugriffs-Datei an und speichert eine Datenliste in den Sätzen der Datei. Achten Sie auf die Verwendung des Parameters L bei der OPEN-Anweisung und des Parameters R bei der WRITE-Anweisung.

```
 10 REM EINE DIREKTZUGRIFFS-DATEI ANLEGEN
 20 D$ = CHR$(4)
 30 DATA AXT,HAMMER,SAEGE,SCHRAUBENZIEHER
 31 DATA ZANGE,SCHRAUBENSCHLUESSEL,HOBEL,SCHRAUB-
    STOCK,NAEGEL
100 HOME
110 VTAB 2: PRINT "LEGE DATEI AN"
120 GOSUB 1000
```

```
1000 PRINT D$;"OPEN DIREKT.DATEN,L50"
1010 FOR A = 1 TO 9
1020 PRINT D$; "WRITE DIREKT.DATEN,R";A
1030 READ B$
1040 PRINT B$
1050 NEXT A
1090 PRINT D$;"CLOSE DIREKT.DATEN"
1099 RETURN
```

Die Funktionsweise dieses Programms ist sehr einfach. Nachdem die Datei in Zeile 1000 mit der Satzlänge 50 geöffnet worden ist, wird die FOR/NEXT-Schleife zwischen den Zeilen 1010 und 1050 neunmal durchlaufen. Nach jedem Durchlauf wird der Positionszeiger durch Zeile 1020 auf einen neuen Satz in der Datei gestellt. ProDOS führt alle Berechnungen zur Umformung der Satznummer in die richtige Bytenummer ab Dateianfang von selbst aus, wenn es den Positionszeiger stellt. Bei Ausführung der Zeile 1030 wird jedesmal eine neue Zeichenkette von den Daten in den Zeilen 30 und 31 in die Variable B$ eingelesen. Zeile 1040 schreibt dann den Inhalt der Variablen B$ auf die Diskette. Hinter der Schleife schließt das Programm die Datei, bevor es endet.

Sie können dieses Programm für die Eingabe beliebiger Daten modifizieren, wenn Sie die Anweisung in Zeile 1030 durch INPUT B$ ersetzen. Danach ignoriert das Programm die DATA-Anweisungen in den Zeilen 30 und 31 und erwartet die Eingabedaten über die Tastatur. Das Programm wird die Schleife der Zeilen 1010 bis 1050 trotzdem ausführen, aber es gibt keinen Grund, diese Werte beizubehalten. Wenn Sie die Sätze 45 bis 50 anstelle der Sätze 1 bis 9 schreiben wollen, können Sie das durch Ändern der Zeile 1010 in

```
1010 FOR A = 45 TO 50
```

tun.

Eine Direktzugriffs-Datei lesen

Wenn Sie eine Direktzugriffs-Datei lesen wollen, müssen Sie in der OPEN-Anweisung den Parameter L benutzen, um ProDOS die Länge eines einzelnen Satzes in der Datei mitzuteilen. Diese Länge wird wieder durch eine Anzahl von Bytes ausgedrückt und muß mit der Satzlänge beim Anlegen der Datei übereinstimmen. Wenn Sie eine andere Satzlänge angeben, werden die Daten möglicherweise unlesbar. (Unter ProDOS können Sie die Satzlänge einer Direktzugriffs-Datei herausfinden, indem Sie sich mit dem CATALOG-Befehl das Inhaltsverzeichnis anschauen.) Beim Lesen einer Direktzugriffs-Datei müssen Sie die Num-

mer des gewünschten Satzes angeben, oder ProDOS liest automatisch die Daten in Satz 0. Verwenden Sie wieder den Parameter R der READ-Anweisung dazu, um die Satznummer anzugeben.

Das folgende Programm öffnet eine Direktzugriffs-Datei und liest einen Satz von der Datei. Die Satznummer wird mit der INPUT-Anweisung angegeben. Geben Sie dort die Zahl 10 ein, wenn Sie das Programm beenden wollen. Beachten Sie den Parameter L bei der OPEN-Anweisung und den Parameter R bei der READ-Anweisung.

```
  10 REM EINE DIREKTZUGRIFFS-DATEI LESEN
  20 D$ = CHR(4)
 100 HOME
 110 VTAB 2: PRINT "LIES EINE DATEI"
 120 VTAB 20: PRINT "GEBEN SIE 10 ZUM BEENDEN EIN"
1000 VTAB 5: INPUT "SATZNR. = ";A%
1010 IF A% < 1 OR A% > 10 GOTO 1000
1020 IF A% = 10 GOTO 1099
1030 PRINT D$,"OPEN DIREKT.DATEN,L50"
1040 PRINT D$,"READ DIREKT.DATEN,R";A%
1060 INPUT B$
1065 VTAB 9: CALL -868: REM LOESCHT BILDSCHIRM
1070 VTAB 9: PRINT "SATZNR.";A%;" = ";B$
1080 PRINT D$;"CLOSE DIREKT.DATEN"
1090 GOTO 1000
1099 END
```

In Zeile 1010 findet eine Überprüfung der eingegebenen Satznummer statt. Ohne diese Anweisung könnte man zum Beispiel den Wert 14 eingeben, für den es keinen entsprechenden Satz in der Datei gibt. Wenn ProDOS auf die Anweisung stößt, auf einen nicht in der Datei vorhandenen Satz zuzugreifen, erzeugt es die Fehlermeldung RANGE ERROR und bricht das Programm ab.

Wenn Sie dieses Programm mit mehr als neun Sätzen in einer Datei benutzen wollen, müssen Sie die Werte in den IF-Tests in den Zeilen 1010 und 1020 ändern. Wenn Sie zum Beispiel 99 Sätze in der Datei zulassen und das Programm mit dem Wert 100 beenden möchten, müssen Sie das Programm folgendermaßen ändern:

```
 120 VTAB 20: PRINT "GEBEN SIE 100 ZUM BEENDEN EIN"
1010 IF A% < 1 OR A% > 100 GOTO 1000
1020 IF A% = 100 GOTO 1099
```

Jetzt können Sie auch größere Werte für A$ eingeben. Wenn Sie jedoch versuchen, auf einen nicht vorhandenen Satz zuzugreifen (d. h. der von ProDOS berechnete Wert für die gegenwärtige Zeigerposition liegt hinter der Markierung des Dateiendes), verursachen Sie eine Fehlermeldung.

Sätze in einer Direktzugriffs-Datei verändern

Direktzugriffs-Dateien haben einen großen Vorteil: Man kann schon existierende Sätze modifizieren. Das ist bei sequentiellen Dateien wegen der variablen Satzlänge sehr schwierig. Bei einer Direktzugriffs-Datei können Sie einfach den gewünschten Satz einlesen, ändern und wieder an dieselbe Stelle innerhalb der Datei zurückschreiben.

Das folgende Programm demonstriert dieses Prinzip. Sie können mit ihm einen beliebigen Satz in der Datei DIREKT.DATEN lesen, modifizieren und auf die Diskette zurückschreiben.

```
  10 REM EINE DIREKTZUGRIFFS-DATEI MODIFIZIEREN
  20 D$ = CHR(4)
 100 HOME
 110 VTAB 2: PRINT "MODIFIZIERE EINE DIREKTZUGRIFFS-
     DATEI"
 120 VTAB 4: PRINT "GEBEN SIE STOP ZUM BEENDEN EIN"
 130 GOSUB 1000
 999 END
1000 PRINT D$;"OPEN DIREKT.DATEN,L50"
1010 VTAB 10: INPUT "SATZNUMMER = ";A%
1020 PRINT D$;"READ DIREKT.DATEN,R";A%
1030 INPUT B$
1040 VTAB 12: PRINT B$
1050 PRINT D$;"WRITE DIREKT.DATEN,R";A%
1060 VTAB 12: INPUT B$
1070 IF B$ = "STOP" THEN GOTO 1200
1080 PRINT B$
1090 GOTO 1010
1200 PRINT D$;"CLOSE DIREKT.DATEN"
1299 RETURN
```

Mit den Pfeiltasten können Sie den Cursor an die Stelle bewegen, wo Sie die Daten verändern wollen, und die alten Daten einfach überschreiben. Vergewissern Sie sich, daß der Cursor am Ende der Zeile steht, bevor Sie die Return-Taste drücken, oder der auf die Diskette zurückgeschriebene Satz wird an der Cursor-Position abgeschnitten.

Sätze an eine Direktzugriffs-Datei anfügen

Das folgende Programm verwendet den APPEND-Befehl dazu, um fünf Sätze an das Ende der Datei DIREKT.DATEN anzufügen. Wenn Sie dieses Programm laufen lassen, sollten sich 14 Sätze in der Datei befinden. (Wenn Sie das frühere Programm zum Einlesen auf einer Diskette gespeichert haben, können Sie es für die zusätzlichen Sätze leicht modifizieren. Setzen Sie den Wert 10 in den Zeilen 120, 1010 und 1020 auf 15. Dann können Sie die Sätze 10 bis 14 einlesen und das Programm durch Eingeben von 15 beenden.)

```
  10 REM AN EINE DIREKTZUGRIFFS-DATEI ANFUEGEN
  20 D$ = CHR$(4)
  30 DATA SCHUHE,HEMD,HOSE,JACKE,SCHLIPS
 100 HOME
 110 VTAB2: PRINT "FUEGE AN EINE DATEI AN"
 120 GOSUB 1000
 999 END
1000 PRINT D$;"APPEND DIREKT.DATEN,L50"
1010 FOR A = 1 TO 5
1030 READ B$
1040 PRINT B$
1050 NEXT A
1090 PRINT D$;"CLOSE DIREKT.DATEN"
1099 RETURN
```

Die APPEND-Anweisung ist sehr nützlich, wenn Sie Sätze ans Ende einer Direktzugriffs-Datei anfügen wollen. Sie werden im obigen Programm bemerken, daß die Schleife, in der die Daten auf die Diskette geschrieben werden, keine WRITE-Anweisung enthält. Wenn Sie im APPEND-Modus sind, brauchen Sie nicht dauernd WRITE- Anweisungen mit den zu verwendenden Satznummern auzugeben. Das liegt daran, daß der Positionszeiger nach dem Übertragen von Daten auf eine Diskette ständig auf den neuesten Stand gebracht wird. Sie sollten jedoch mit der früher besprochenen Technik überprüfen, ob diese Daten die richtige Länge haben. Tun Sie das nicht, passen die auf die Diskette übertragenen Daten möglicherweise nicht in einen Satz, und Sie können von einem späteren Programm nicht mehr richtig gelesen werden. Sie können dann jedoch die Datei als sequentielle Datei behandeln, wenn Sie die Daten zurückgewinnen wollen. Zu diesem Zweck müssen Sie die Datei öffnen und sie vom Anfang bis zum Ende sequentiell durchlesen.

DIE EXEC-ANWEISUNG UND TEXTDATEIEN

Es gibt bei ProDOS noch eine weitere mächtige Anweisung zur Behandlung von Textdateien: die EXEC-Anweisung. Sie kann zur Kontrolle der Ausführung von allgemeinen Aufgaben oder individuellen Programmen von einer Datei aus verwendet werden.

Die EXEC-Anweisung veranlaßt Ihren Apple, Eingabedaten von einer Diskettendatei anstatt von der Tastatur zu nehmen. Diese Diskettendatei kann eine Folge von Anweisungen enthalten, wie RUN PROGRAMM1 oder DELETE MEINEDATEI, oder sie kann eine Ansammlung von Daten enthalten. Daten und Anweisungen können auch kombiniert werden.

Bevor wir weitermachen, sind ein paar Definitionen angebracht. Eine EXEC-Anweisung ist ein Befehl, der genauso wie alle anderen ProDOS-Befehle verwendet wird. Eine EXEC-Datei ist eine Textdatei, die Daten oder Befehle enthält, die bei der EXEC- Anweisung verwendet werden. Ein EXEC-Programm ist eine EXEC-Datei, die Applesoft- oder ProDOS-Befehle enthält, im Gegensatz zu einer, die nur Daten für Eingabezwecke enthält.

Die bei der EXEC-Anweisung verwendete Datei kann sowohl eine Direktzugriffs-Datei als auch eine sequentielle Datei sein; sie wird jedoch immer als sequentielle Datei behandelt (sie wird der Reihe nach von Anfang bis Ende durchgelesen). Daran ändert sich auch dadurch nichts, daß Sie EXEC-Dateien als solche bezeichnen können, um auszudrücken, daß es sich nicht um Standarddateien handelt.

Wird eine EXEC-Anweisung verwendet, bleibt sie in Kraft, bis das Ende der in diesem Befehl angegebenen Datei erreicht worden ist. Das bedeutet: Sie können ein Applesoft-Programm, das unter einem EXEC-Befehl läuft, nicht mit Control-C unterbrechen, weil der Apple die Eingabe nicht von der Tastatur, sondern von der EXEC- Datei erwartet. Aus demselben Grund können Sie die Ausführung der Befehle in einem EXEC-Programm mit Control-C auch nicht unterbrechen, wenn es sich nicht bei der Ausführung eines Applesoft-Programms befindet. Sie müssen auf drastischere Maßnahmen wie das Ausschalten des Computers zurückgreifen, wenn Sie ein EXEC-Programm stoppen wollen.

Sie können den Parameter F dazu verwenden, eine Anzahl von Sätzen am Anfang der Datei zu überspringen. Wenn Sie zum Beispiel bei der Ausführung der Datei EXEC1.EXE die ersten beiden Anweisungen auslassen wollen, können Sie die Anweisung

```
EXEC EXEC1.EXE,F2
```

verwenden.

Den EXEC-Befehl können Sie auf verschiedene Weise benutzen. Sie können zum Beispiel den EXEC-Befehl mit einer Datei verwenden, um die gesamte Eingabe in ein Programm zu kontrollieren. Das kann beim Testen und Berichtigen eines Programms sehr nützlich sein, weil Sie damit eine Standardmenge von Eingabedaten für den Test beibehalten können. Dadurch wird das Tippen von Fehlern verhindert und Zeit beim Wiederholen des Tests gespart.

Sie können den EXEC-Befehl dazu verwenden, vom BASIC-Prompt aus eine Serie von Befehlen auszugeben. So können aus vielen Schritten zusammengesetzte Standardprozeduren gebildet und in einer Diskettendatei gespeichert werden. Danach können Sie eine solche Prozedur beliebig oft wiederholen, indem Sie den EXEC- Befehl mit der Datei eingeben, in der die Prozedur gespeichert ist.

Mit dem EXEC-Befehl können Sie auch Programme verschmelzen. Dann brauchen Sie nicht für jedes Programm die gleichen Programmzeilen einzugeben. Sowohl Teilprogramme als auch ganze Programme können verschmolzen werden.

Beispiele für diese Funktionen werden in den folgenden Abschnitten gegeben.

Eine EXEC-Datei anlegen

Das folgende Programm erzeugt eine sequentielle Datei, die die Anweisungen enthält, die Sie eingeben. Wenn Sie alle Anweisungen eingetippt haben, die Sie in der Datei haben wollen, geben Sie das Wort STOP ein. Ihre Anweisungen oder Daten werden dann in der Datei für den späteren Gebrauch gespeichert.

```
  50 DIM B$(50)
 100 D$ = CHR$(4)
 110 GOSUB 1000
 120 GOSUB 2000
 999 END
1000 HOME: PRINT "EXEC-EINGABEPHASE": VTAB 4
1005 A = 0
1010 A = A + 1
1020 PRINT A;
1040 INPUT " ";B$(A)
```

```
1050 IF B$(A) = "STOP" GOTO 1090
1060 GOTO 1010
1090 B = A
1095 B$(A) = ""
1099 RETURN
2000 HOME: PRINT "EXEC-DATEIANLEGEPHASE"
2010 PRINT D$;"OPEN EXEC1.EXE"
2020 PRINT D$;"WRITE EXEC1.EXE"
2030 FOR A = 1 TO B
2040 PRINT B$(A)
2050 NEXT A
2060 PRINT D$;"CLOSE EXEC1.EXE"
2099 RETURN
```

Mit dem ersten Teil des Programms können Sie bis zu 50 Zeilen Text in das Feld B$ eingeben. Brauchen Sie noch mehr, müssen Sie die DIM-Anweisung in Zeile 50 ändern. Wenn Sie die Zeichenkette STOP eingeben, endet das erste Unterprogramm, und alle eingegebenen Zeilen werden im zweiten Unterprogramm in die Datei EXEC1.EXE geschrieben. Zeile 1095 entfernt das STOP aus dem Feld, so daß es nicht in die Datei geschrieben wird.

Wenn Sie die drei Zeilen

```
CAT
DELETE DIREKT.DATEN
CAT
```

in die Datei EXEC1.EXE eintippen und dann die Anweisung EXEC EXEC1.EXE eingeben, wird jede dieser Anweisungen einzeln ausgeführt. Zuerst sehen Sie, wie ProDOS den CAT-Befehl ausführt und Ihnen alle Dateien auf der Diskette zeigt. Dann führt ProDOS die DELETE-Anweisung aus (Voraussetzung ist, eine Datei mit dem Namen DIREKT.DATEN befindet sich auf der Diskette). Schließlich führt ProDOS die zweite CAT-Anweisung aus und zeigt Ihnen wieder die Dateien auf der Diskette, wo DIREKT.DATEN jetzt fehlt.

Sie können mit dieser Technik eine Serie von Programmen laufen lassen, indem Sie eine Reihe von RUN-Befehlen als Zeilen in eine EXEC-Datei eintippen und dann die EXEC-Anweisung auf diese Datei anwenden.

Eingabedaten über EXEC

EXEC bietet eine weitere interessante Möglichkeit: Sie können Eingabedaten in eine Datei schreiben und dann dazu verwenden, den Ablauf

eines Programms zu kontrollieren. Sie können das vorherige Programm dazu benutzen, eine Datei mit numerischen Daten anstelle von Anweisungen anzulegen, und dann das folgende kleine Programm eintippen und ausführen, um zu sehen, wie das in der Praxis funktioniert. Diese Technik wird häufig für Demonstrations- und Übungszwecke benutzt.

```
 1 REM TEST VON EXEC
 5 D$ = CHR$(4)
10 ONERR GOTO 99
20 PRINT D$;"EXEC EXEC1.EXE"
30 INPUT A
40 INPUT B
50 INPUT C
55 D = C * (A/B)
60 PRINT "D = ";C;" * (";A;"/";B;")"
70 PRINT "D = ";D
80 GOTO 30
99 PRINT D$;"CLOSE EXEC1.EXE"
```

Wenn Sie die obige Datei mit zehn Textzeilen angelegt haben und diese Zeilen die Zahlen 5, 10, 12, 15, 16, 17, 21, 22, 23 und 50 enthalten, führt das Programm drei vollständige Schleifen aus. Beim ersten Durchlauf sind die Werte für A, B und C gleich 5, 10 und 12. Beim zweiten und dritten Durchlauf der Schleife werden jeweils die drei nächsten Zahlen genommen. Beim vierten Durchlauf erhält A den Wert 50. Wenn aber die INPUT-Anweisung für B in Zeile 40 ausgeführt wird, ist das Dateiende erreicht, und die ONERR-Anweisung in Zeile 10 wird aktiviert. Diese veranlaßt, daß die Datei geschlossen und das Programm beendet wird.

Der Datentyp, den die EXEC-Anweisung von der Datei liest, muß mit dem Variablentyp in der INPUT-Anweisung übereinstimmen; sonst entsteht ein Fehler. Wenn das erste Datenelement in der Datei nicht 5, sondern CAT wäre, würde bei der Ausführung von Zeile 30 ein Fehler entstehen. Die Zeichenkette CAT darf nicht in die numerische Variable A eingesetzt werden.

Mit EXEC Programme kopieren

Oft werden Sie feststellen, daß Sie einen großen Teil der Programmzeilen, die Sie für ein neues Programm brauchen, schon in anderen Programmen geschrieben haben. Das ist besonders dann der Fall, wenn Sie sich an modulare Programmierung halten und einmalige Unterprogramme für allgemeine Funktionen schreiben. Sie können mit der EXEC-Anweisung

die alten Zeilen in Ihr neues BASIC-Programm kopieren und sich damit eine Menge Arbeit ersparen.

Dazu müssen Sie zuerst eine sequentielle Textdatei anlegen, die die Programmzeilen enthält, die Sie kopieren wollen. Das folgende Unterprogramm kann zu diesem Zweck in jedes Programm eingefügt und mit der Anweisung RUN 63500 ausgeführt werden. (Diese Anweisung beginnt mit der Ausführung des Programms in Zeile 63500.) Die bei der LIST-Anweisung angegebenen Zeilennummern können geändert werden, so daß nur die Zeilen eingefügt werden, die Sie kopieren möchten.

```
63500 D$ = CHR$(4)
63506 INPUT "DATEI:";XX$
63508 PRINT D$;"OPEN";XX$
63510 PRINT D$;"WRITE";XX$
63512 LIST 1,63499
63514 PRINT D$;"CLOSE";XX$
63516 END
```

Wenn dieses Unterprogramm ausgeführt wird, veranlaßt Zeile 63512, daß alle Programmzeilen von 1 bis 63499 in eine sequentielle Datei geschrieben werden. In der Variablen XX$ wird der Dateiname gespeichert, den Sie in Zeile 63506 eingeben. Das ist der Name der Ausgabedatei.

Nun müssen Sie die EXEC-Anweisung benutzen, um diese Zeilen in Ihr Programm einzufügen. Das geht, indem Sie Ihr neues Programm in den Speicher laden und EXEC zusammen mit dem Dateinamen eingeben, unter dem Sie die Programmzeilen gespeichert haben. Die Datei besteht aus Befehlen mit Zeilennummern, die folglich nicht direkt ausgeführt, sondern nur zusammen mit dem übrigen Programm im Speicher abgelegt werden.

Wenn Sie zum Beispiel die Zeilen 500 bis 1000 Ihres Programms in einer Datei namens CODE500 und Ihr neues Programm in einer Datei namens NEUERCODE gespeichert haben, können Sie diese beiden Dateien folgendermaßen verschmelzen. Zuerst geben Sie den Befehl LOAD NEUERCODE ein. Damit wird Ihr neues Programm in den Speicher geladen. Als zweites geben Sie den Befehl EXEC CODE500 ein. Damit wird der Inhalt der Dateien CODE500 und NEUERCODE im Speicher verschmolzen. Als drittes speichern Sie das Programm, das sich jetzt im Speicher befindet, unter einem beliebigen Namen ab (z. B. mit dem Befehl SAVE VERSCHMOLZEN).

Seien Sie sich darüber im klaren: Jede Zeile in der EXEC-Datei, die die gleiche Zeilennummer wie eine Codezeile im Speicher hat, ersetzt diese beim Einlesen der EXEC-Datei.

NÜTZLICHE HINWEISE

Es folgen ein paar Hinweise und Vorschläge zum Programmieren mit Textdateien.

Die Anweisungen READ und WRITE bleiben bis zur Ausgabe eines neuen ProDOS-Befehls in Kraft. Oft möchten Sie sofort nach dem Einlesen von Daten aus einer Diskettendatei etwas über die Tastatur eingeben und auf dem Bildschirm darstellen lassen. Eine ähnliche Situation entsteht bei der Anwendung des PRINT-Befehls hinter einer WRITE-Anweisung. Damit können Sie leicht fertig werden, indem Sie ein GOSUB zu einem Unterprogramm wie

```
PRINT D$:PRINT:RETURN
```

ausführen lassen. Was zur Aufhebung der Wirkung einer READ- oder WRITE-Anweisung wirklich benötigt wird, ist das Control-D-Zeichen, nicht der eigentliche ProDOS-Befehl. Dieses Unterprogramm dient dazu, ProDOS mit einem Control-D-Zeichen zu aktivieren, ohne daß ein ProDOS-Befehl ausgeführt wird.

Bei Direktzugriffs-Dateien müssen Sie die Satzlänge einer Datei nachhalten. Handgeschriebene Notizen verschwinden oft, wenn man sie am nötigsten braucht. Viele Programmierer fügen die Satzlänge in den Dateinamen ein, um sie nicht zu verlieren. Andere benutzen den Satz 0 der Datei als leicht zugängliche Kopfzeile für Angaben über die Satzlänge oder die Anzahl der Datenelemente in einem Satz. Probieren Sie diese Methoden aus, um zu sehen, was für Sie geeignet ist.

Diese Technik ist jetzt nicht mehr so wichtig wie bei DOS 3.3, weil Sie sich bei ProDOS die Satzlänge einer Direktzugriffs-Datei mit dem CATALOG-Befehl ausgeben lassen können. Sie kann dennoch nützlich sein, um die Kompatibilität mit früheren Versionen Ihres Programms sicherzustellen oder zur Dokumentation, wenn Sie solche Dateien in das ältere Betriebssystem DOS 3.3 übertragen wollen.

UNTERSCHIEDE ZU DOS 3.3

Die wichtigste Verbesserung bei der Handhabung von Dateien durch ProDOS beruht darauf, daß Sie die Parameter F und B bei den READ- und

WRITE-Befehlen benutzen können. Wenn Sie unter DOS auf eine bestimmte Stelle innerhalb einer Datei zugreifen wollten, mußten Sie entweder eine Schleife zum Lesen der Datei schreiben oder die separate POSITION-Anweisung benutzen. Wegen dieser zusätzlichen Möglichkeit ist die POSITION-Anweisung überflüssig geworden, obwohl sie aus Gründen der Kontinuität zu DOS beibehalten wurde.

Der zweite größere Unterschied zwischen DOS und ProDOS liegt in der Behandlung von Direktzugriffs-Dateien. DOS 3.3 speicherte alle Textdateien auf die gleiche Weise ab und trug die Satzlänge einer jeden Datei nicht im Inhaltsverzeichnis ein. Deshalb können Sie keine Direktzugriff-Dateien unter DOS in Direktzugriffs-Dateien unter ProDOS übertragen. Sie können jedoch eine Direktzugriffs-Datei unter DOS als sequentielle Datei übertragen und dann ein kurzes Programm schreiben, das die sequentielle Datei liest und sie als Direktzugriffs-Datei im ProDOS-Format auf die Diskette zurückschreibt.

Kapitel 8

ProDOS und binäre Dateien

In diesem Kapitel steht die Verwendung von binären Dateien unter ProDOS im Vordergrund. ProDOS behandelt diese Dateien ähnlich wie Textdateien; da eine binäre Datei jedoch Daten jedes beliebigen Typs enthalten kann, müssen bei binären Dateien spezielle Befehle verwendet werden. Das sind die Befehle BRUN, BLOAD und BSAVE. Wir werden diese Befehle auf den folgenden Seiten besprechen.

BINÄRE DATEIEN

Technisch gesehen sind alle Dateien binäre Dateien. Dazu gehören sowohl Textdateien und BASIC-Programme als auch die Dateien, die beim Auflisten mit dem CATALOG-Befehl mit .BIN gekennzeichnet sind; denn der Computer speichert alle Daten binär. ProDOS-Dateitypen wie TXT, BAS und BIN (siehe Anhang A) stellen eine andere Einteilungsstufe dar. Textdateien (Typ TXT) sind beispielsweise Dateien, die nach speziellen Regeln behandelt werden, wenn man auf sie mit einem ProDOS-Befehl zugreift. Die BASIC-Anweisung INPUT interpretiert das Return-Zeichen (ASCII 13) in einer Textdatei als Ende eines Satzes.

Bei ProDOS definieren wir binäre Dateien als Dateien vom Typ BIN. Diese können nur mit den Befehlen BSAVE, BLOAD und BRUN behandelt werden. Binäre Dateien können mit diesen Befehlen direkt vom Speicher auf eine Diskette geschrieben oder von einer Diskette in den Speicher geladen werden. Sie können sogar festlegen, welchen Bereich des Speichers Sie auf die Diskette schreiben wollen oder an welche Stelle Sie die Daten im Speicher laden wollen. Das kann ein wirkungsvolles Hilfsmittel bei der Behandlung von Grafik- oder von Maschinensprache-Programmen sein. Damit können Sie die Funktionsweise Ihres Computers mit einer Genauigkeit und einer Präzision kontrollieren, wie das unter normalen Umständen nicht möglich ist.

Binäre Dateien werden zum Beispiel häufig dazu verwendet, Unterprogramme in Maschinensprache von der Diskette in den Speicher zu laden,

damit sie von einem BASIC-Programm aus aufgerufen werden können. Das BASIC-Programm muß genau wissen, wo sich das entsprechende Unterprogramm befindet, und das Unterprogramm muß in einem Speicherbereich sein, der nicht von anderen Teilen des Programms benutzt wird. In diesem Fall ist es wichtig, eine Datei direkt von der Diskette an eine bestimmte Stelle im Speicher laden zu können.

Ein weiterer großer Unterschied zwischen binären Dateien und anderen Dateitypen besteht darin, daß bei den Befehlen BLOAD, BSAVE und BRUN nicht versucht wird, den Inhalt der zu übertragenden Dateien zu interpretieren. Das bedeutet: Ein Zeichen wie ASCII 13 wird genau wie jedes andere Zeichen behandelt und nicht als Markierung eines Satzendes.

Diese Eigenschaft, Daten nicht zu interpretieren, ermöglicht es ProDOS, Maschinensprache-Programme fehlerfrei in binären Dateien zu speichern. Sie können so auch Grafikbilder abspeichern und sie später wieder in den Grafikbereich zurückübertragen. Das sind die beiden häufigsten Verwendungszwecke für binäre Dateien.

ProDOS-BEFEHLE BEI BINÄREN DATEIEN

ProDOS stellt drei Anweisungen zur Behandlung von binären Dateien zur Verfügung: BSAVE, BLOAD und BRUN. Mit diesen Befehlen können Sie binäre Daten auf eine Diskette speichern, binäre Dateien von einer Diskette in den Speicher laden oder binäre Dateien ausführen, wenn es sich um Programmdateien handelt. Sie sind im wesentlichen den Anweisungen SAVE, LOAD und RUN bei BASIC-Programmen gleich, aber BSAVE und BLOAD funktionieren bei jedem Dateityp. Den BRUN-Befehl kann man nur auf Dateien vom Typ BIN mit Maschinensprache-Programmen anwenden. Um diese Befehle von den entsprechenden BASIC-Befehlen zu unterscheiden, wird ihnen der Buchstabe B vorangestellt. Das kennzeichnet sie als Befehle für binäre Dateien.

Die Befehle BSAVE, BLOAD und BRUN gab es schon unter DOS 3.3, es sind jedoch einige Verbesserungen vorgenommen worden, die sie flexibler und mächtiger machen. Sie können jetzt zum Beispiel auch einen Teil einer binären Datei laden und ausführen; Sie können zwei verschiedene Methoden benutzen, um die Länge einer Datei festzulegen, und BLOAD und BSAVE können jetzt bei jedem beliebigen Dateityp verwendet werden, nicht nur bei binären Dateien.

Auch mit dem Strich-Befehl (–) können Sie versuchen, eine binäre Datei

laufen zu lassen. Dieser Befehl funktioniert bei binären Dateien genau wie bei Dateien vom Typ BAS, SYS und EXE. (SYS- Dateien werden bei diesem Befehl als BIN-Dateien behandelt; sie sind Programme des Betriebssystems wie z. B. BASIC.SYSTEM.) Beim Strich-Befehl gibt es jedoch keine von den Adreßoptionen, die wir in diesem Kapitel besprechen werden. Bei der verbesserten Verzeichnisstruktur von ProDOS wird die Speicheradresse, von der aus eine Datei auf die Diskette übertragen wurde (mit BSAVE), in einem Verzeichniseintrag festgehalten. Wenn Sie beim BRUN- oder beim Strich-Befehl keine Ladeadresse angeben, benutzt ProDOS diese Adresse als Voreinstellungswert. Das kann Ihnen viel Zeit und Ärger bei der Ausführung eines Programms ersparen.

Ein Wort der Vorsicht sei zum BRUN- und zum Strich-Befehl gesagt. Der BRUN-Befehl überprüft, ob es sich um eine Datei vom Typ BIN handelt, aber er kann nicht feststellen, ob die Datei ein reguläres Programm enthält. Da eine BIN-Datei alles Mögliche enthalten kann, von einem Programm in Maschinensprache über Grafikdaten bis zu numerischen Daten oder sogar zufälligen Daten von früheren Programmen, gibt es keine Garantie dafür, daß die Datei, die Sie mit BRUN aufrufen, auch wirklich läuft. Wenn die Datei ein gültiges Programm enthält, trifft dies zu. Andernfalls ist ungewiß, ob die Datei lauffähig ist. Grafikdaten werden von der BRUN-Anweisung als Maschinensprache-Befehle interpretiert, und einige der Grafikdaten könnten sich zufällig zu einer Befehlsfolge zusammensetzen, die der Apple ausführen kann. Dabei kann eine Folge von Syntaxfehlern erzeugt werden, das System kann abstürzen und muß eventuell neu geladen werden. Es ist sogar möglich – wie bei dem Beispiel des Affen, der eine Million Jahre auf einer Schreibmaschine herumhämmert und dabei zufällig ein Werk von Shakespeare reproduziert –, daß ein gültiges und sinnvolles Programm gefunden wird.

Das gleiche gilt für den Strich-Befehl, obwohl dieser Befehl noch nicht einmal den Dateityp überprüft. Experimentieren Sie selbst damit. Versuchen Sie mit dem Strich-Befehl eine Datei laufen zu lassen, die eigentlich nicht laufen dürfte, wie zum Beispiel eine Textdatei. Probieren Sie es bei mehreren Dateien, und beobachten Sie, wie verschieden die Ergebnisse sind.

Genau das meinen wir, wenn wir sagen, Sie können „versuchen", eine Datei laufen zu lassen. Die Befehle BRUN und – laden die Datei und beginnen mit der Ausführung der ersten Instruktion (dem ersten Byte der Datei). Wenn der Inhalt der Datei keine gültige Folge von Befehlen darstellt, kann Ihr Apple die Ausführung dieses „Programms" nicht erfolgreich beenden.

Die BSAVE-Anweisung

Bei der BSAVE-Anweisung wird die angegebene Anzahl von Bytes direkt vom aktiven Speicher Ihres Apple in die Diskettendatei geschrieben, die Sie in dem Befehl angeben. Falls diese Datei noch nicht auf der Diskette existiert, wird eine neue angelegt. Der Inhalt einer bereits vorhandenen Datei mit dem von Ihnen angegebenen Pfadnamen wird zerstört und durch die neuen Daten ersetzt. Wenn Sie bei dieser Anweisung den Parameter T benutzen, können Sie jeden beliebigen Dateityp vom Speicher auf die Diskette übertragen. Andernfalls können Sie nur binäre Dateien abspeichern. Jeder Versuch, eine nicht binäre Datei abzuspeichern, verursacht dann die Fehlermeldung FILE TYPE MISMATCH (falscher Dateityp).

Bei der BSAVE-Anweisung muß der Parameter A (der die Startadresse angibt) und entweder der Parameter L oder E (die beide die letzte Speicheradresse angeben) benutzt werden. Dann kann ProDOS den Startpunkt im Speicher und die Anzahl der Bytes bestimmen, die auf die Diskette übertragen werden sollen. Sie können den in Kapitel 7 besprochenen Parameter B verwenden und damit das erste Byte innerhalb der Datei, in die Sie schreiben wollen, angeben, so daß ProDOS weiß, wo es innerhalb der Datei beginnen soll.

Wenn Sie diese Parameter nicht angeben, können zwei verschiedene Dinge passieren. Wenn Sie mit BSAVE eine Datei abspeichern wollen, die noch nicht auf der Diskette existiert, liest ProDOS das Inhaltsverzeichnis der Diskette und kommt mit der Fehlermeldung

```
PATH NOT FOUND
```

zurück. Wenn die Datei schon existiert, liest ProDOS das Inhaltsverzeichnis, um sich über die Startadresse und die Länge der Datei zu informieren, und überträgt die Datei dann aus dem Speicher auf die Diskette. Das hat gewisse Vorteile. Wenn Sie es zum Beispiel mit Grafikdateien zu tun haben, werden sich die Startadresse und die Dateilänge kaum ändern. Die Möglichkeit, nur BSAVE und den Dateinamen einzugeben, kann Ihnen einiges Herumhantieren mit dem Handbuch, um herauszufinden, wo der Grafikbereich anfängt, ersparen, obwohl Sie beim ersten Abspeichern der Datei die Parameter A, L und E immer noch angeben müssen. Der Nachteil besteht darin, daß sich Ihr Programm oder Ihre Daten genau an der richtigen Stelle befinden müssen, damit der Befehl korrekt ausgeführt werden kann. Wenn Sie Ihre Programmzeilen im Speicher verschoben haben, vielleicht weil Sie einen Bereich zur Entwicklung und einen anderen Bereich zur Ausführung benutzen, bleibt im Dateieintrag des

Inhaltsverzeichnisses die Adresse stehen, die Sie angegeben haben, als Sie zuletzt mit dem vollständigen BSAVE-Befehl abgespeichert haben – BSAVE plus Pfadname, Parameter A und L oder E. Wenn Adresse und Länge nicht mehr mit dem Bereich übereinstimmen, den Sie abspeichern wollen, werden Ihre Daten auf der Diskette durch die Daten ersetzt, die sich gerade in dem Speicherbereich befinden, aus dem Sie die Datei zuletzt mit dem vollständigen Befehl abgespeichert haben.

Der BSAVE-Befehl untersucht beim Abspeichern nicht, welcher Datentyp vorliegt. Er funktioniert mit jedem gültigen Speicherbereich, solange zum Abspeichern dieses Bereichs genügend Platz auf der Diskette vorhanden ist. Die Daten brauchen kein gültiges Programm zu bilden; sie brauchen nicht einmal aus codierten Anweisungen irgendeiner Form zu bestehen. Deshalb können Sie auch Bilder von den Grafikseiten des Apple abspeichern und diese später schnell und genau mit der BLOAD-Anweisung wieder zurückholen.

Die BLOAD-Anweisung

Die BLOAD-Anweisung wird dazu verwendet, den Inhalt einer Datei direkt von der Diskette in den Speicher zu übertragen. Wenn Sie den Parameter T bei dieser Anweisung benutzen, können Sie jeden beliebigen Dateityp übertragen. Andernfalls können Sie nur binäre Dateien laden. Bei jedem Versuch, eine nichtbinäre Datei zu laden, erscheint dann die Fehlermeldung FILE TYPE MISMATCH (falscher Dateityp).

Die BLOAD-Anweisung verlangt den Parameter A (die Startadresse im Speicher), wenn es sich nicht um eine binäre Datei handelt. Die Adresse, von wo eine Datei vom Typ BIN mit BSAVE abgespeichert worden ist, wird nämlich in einem Verzeichniseintrag festgehalten. Wenn bei der BLOAD-Anweisung keine Adresse angegeben wird, wird die Adresse im Verzeichnis als Voreinstellungswert genommen. Ist eine Datei aber nicht mit BSAVE abgespeichert worden, ist kein solcher Eintrag im Verzeichnis gemacht worden. Deshalb müssen Sie den Parameter A benutzen, wenn Sie BLOAD auf nichtbinäre Dateien anwenden.

Mit den Parametern E oder L können Sie die Anzahl der Bytes angeben, die Sie von der Datei lesen wollen. Auf diese Weise können Sie eine binäre Datei als Bibliothek behandeln. Sie können zum Beispiel eine Anzahl verschiedener Routinen in Maschinensprache in einer einzigen binären Datei abspeichern. Dann können Sie den vorhandenen Speicherplatz effizienter ausnutzen, indem Sie nur die Routinen einlesen, die Sie gerade brauchen. Der in Kapitel 7 besprochene Parameter B kann in die-

sem Zusammenhang benutzt werden, wenn eine bestimmte Anzahl von Bytes in der Datei übersprungen werden soll.

Die BLOAD-Anweisung kann sich als sehr gefährlicher Befehl herausstellen, weil das Betriebssystem keinen Versuch macht, Sie am Überschreiben eines bestimmten Speicherbereichs zu hindern. Das bedeutet, daß Sie Ihr System zum Absturz bringen können oder daß Sie die Daten aus der binären Datei an Stellen schreiben können, wo bereits wichtige Daten gespeichert sind. Wenn Sie das Programm oder die Daten, die Sie zerstört haben, nicht vorher auf eine Diskette gerettet haben, sind sie für immer verloren. Sie sollten alle neuen Programme oder empfindlichen Daten stets sorgfältig abspeichern, bevor Sie die BLOAD-Anweisung eingeben. Sie tun auch gut daran, Ihre Berechnungen zur Bestimmung der korrekten Speicherstellen vor der Ausführung dieses Befehls noch einmal zu überprüfen.

Die BRUN-Anweisung

Die BRUN-Anweisung wird zum Laden und Ausführen einer binären Datei benutzt. Sie ist in der Praxis äquivalent zu einer BLOAD- Anweisung mit anschließender Übergabe der Kontrolle an die Anweisung an der Startadresse des Bereichs, in den das Programm geladen worden ist. Aus diesem Grund stimmen die Parameterlisten bei der BRUN- und der BLOAD-Anweisung fast völlig überein. Ausgenommen ist der Parameter T, der bei der BLOAD-Anweisung überflüssig ist, da diese nur auf binäre Dateien (Typ BIN) angewendet werden kann.

Obwohl Sie bei der BRUN-Anweisung vor der Ausführung nicht binärer Dateien geschützt werden, hindert Sie keiner daran, eine binäre Datei auszuführen, die kein gültiges Maschinensprache-Programm enthält. Wenn Sie zum Beispiel die BRUN-Anweisung auf eine binäre Datei anwenden, die ein Bild von der Grafikseite des Apple enthält, kann alles Mögliche passieren. Es kann sofort ein Fehler auftreten, der das Programm zum Stoppen bringt, oder noch schlimmer, es könnte zufällig eine Reihe von gültigen Befehlen auftreten, die den Apple veranlassen, irgend etwas zu tun – etwas Unvorhersehbares, das u. U. das Löschen einer Datei von der Diskette bewirkt. Um das Risiko der Ausführung von Dateien vom Typ BIN, die kein Programm enthalten, zu vermindern, verwenden viele Programmierer sinnfällige Namen für ihre Dateien. Apple schlägt zum Beispiel vor, Dateien, die von den Grafikseiten abgespeicherte Bilder enthalten, stets die Endung .PIC (picture) zu geben. So können Sie auf Anhieb bestimmen, ob eine Datei ein Programm oder Daten enthält. Die Anweisung

```
BSAVE TEST1.PIC,A8192,E16383
```

schreibt z.B. eine Datei vom Typ BIN mit dem Namen TEST1.PIC auf die Diskette. Sie können sie leicht an der Endung PIC als Grafikdatei erkennen, wenn Sie sich diese Datei mit einem der Befehle CAT oder CATALOG anschauen.

Die Speicheradreß-Optionen und der Parameter T

Zusätzlich zu den Parametern für Steckplatz und Laufwerk können bei allen drei binären Befehlen einige oder alle der folgenden Speicheradreß-Optionen verwendet werden. Jede von ihnen hat eine besondere Bedeutung, wenn sie an einen binären Befehl angefügt wird.

Die Option A kann bei den Befehlen BLOAD und BRUN verwendet werden; beim BSAVE-Befehl muß sie angegeben werden. Dieser Parameter gibt die Startadresse des Speicherbereichs an, aus dem Sie eine Datei abspeichern oder in den Sie eine Datei laden wollen. Der Wert hinter dem Buchstaben A kann entweder in dezimaler oder in hexadezimaler Form angegeben werden. Die folgenden beiden Anweisungen sind zum Beispiel identisch:

```
BSAVE TEST,A8200,E8300
BSAVE TEST,A$2008,E8300
```

Beide Anweisungen veranlassen, daß der Speicherbereich von Adresse 8200 bis zur Adresse 8300 mit dem BSAVE-Befehl in eine Diskettendatei übertragen wird. Beachten Sie, daß die zweite Anweisung den einen Wert in dezimaler und den anderen in hexadezimaler Form enthält. Sie werden beide richtig ausgewertet.

Der Parameter E gibt die Adresse des letzten zu übertragenden Bytes an. Dieser Wert muß größer als der beim Parameter A angegebene Wert sein. Er ist einschließlich zu verstehen, d. h., das Byte, das an der Stelle gespeichert ist, die im Parameter E angegebenen ist, wird mitübertragen. Mit den Parametern E und A können Sie Datenblöcke vom und zum Speicher übertragen. Wenn Sie zum Beispiel genau 100 Bytes ab Speicherstelle 8192 in den Speicher laden wollen, können Sie das mit der folgenden Anweisung tun:

```
BSAVE TEST,A8192,E8291
```

Mit dem Parameter L können Sie eine Länge in Bytes angeben. Das ist nur eine andere Art, den Parameter E zu betrachten. Der Parameter L enthält die Anzahl der zu übertragenden Bytes, während mit dem Para-

meter E das letzte zu übertragende Byte angegeben wird. Der Wert des Parameters L ist immer gleich der Endadresse (E) minus der Startadresse (A) plus 1. Die folgenden beiden Anweisungen sind zum Beispiel äuquivalent:

```
BSAVE TEST,A8192,E8291
BSAVE TEST,A8192,L100
```

Beide Anweisungen veranlassen, daß der Speicherbereich von Adresse 8192 bis Adresse 8291 mit dem BSAVE-Befehl auf eine Diskette übertragen wird.

Wenn Sie die Befehle BSAVE oder BLOAD bei nichtbinären Dateien verwenden wollen, können Sie den Parameter T angeben. Diesen Parameter gibt es nicht beim BRUN-Befehl. Auf den Buchstaben T folgen drei Zeichen, die den Dateityp bezeichnen, so daß ProDOS sieht, Sie wollen eine Datei verwenden, die nicht vom Typ BIN ist. Die folgende Anweisung kann zum Beispiel dazu benutzt werden, das BASIC-Programm SEQ.ANLEGEN an die Stelle 3000 im Speicher zu laden:

```
BLOAD SEQ.ANLEGEN,A3000,TBAS
```

Wenn Sie den BSAVE-Befehl bei einer nichtbinären Datei verwenden wollen, deren Name es auf der Diskette noch nicht gibt, müssen Sie diese Datei zuerst mit dem ProDOS-Befehl CREATE unter Angabe des Parameters T mit dem geeigneten Dateityp anlegen. Die folgenden beiden Anweisungen legen zum Beispiel eine Datei TEST vom Typ $F1 auf der Diskette an und füllen sie mit Daten aus dem Speicher:

```
CREATE TEST,T$F1
BSAVE TEST,A8192,L100,T$F1
```

In Anhang A sind alle gültigen ProDOS-Dateitypen aufgelistet. Die nur anwenderdefinierten Dateitypen beginnen mit $F und müssen mit einer Zahl von eins bis acht enden.

Warum möchten Sie die Befehle BLOAD und BSAVE auf nichtbinäre Dateien anwenden? Die Antworten darauf sind so vielfältig wie Ihre Vorstellungskraft. Die Dateien können Daten enthalten, die Sie an bestimmter Stelle im Speicher für ein Programm in Maschinensprache benötigen, oder Daten, die Sie ausgegeben haben möchten. Sie wollen möglicherweise für bestimmte Aufgaben Ihre eigenen Dateitypen schaffen. Sie möchten vielleicht bestimmte Dateitypen in einer Art verwenden, die ProDOS gewöhnlich nicht zuläßt, wie zum Beispiel das Lesen und Schreiben von BASIC-Programmen innerhalb eines anderen Programms, wie

das bei einem Texteditor der Fall ist. Die Möglichkeit, bei ProDOS auf jede beliebige Datei mit den Befehlen BLOAD und BSAVE zugreifen zu können, gibt Ihnen die Freiheit, diese und andere Dinge zu tun.

Bei binären Befehlen können Sie auch den im letzten Kapitel besprochenen Parameter B benutzen. Die folgende Anweisung überspringt zum Beispiel die ersten 50 Bytes einer Datei beim Laden in den Speicher an die Stelle 8192:

```
BLOAD TEST,A8192,B50
```

Dieser Parameter kann auch beim Schreiben mit dem BSAVE-Befehl zum Überspringen von Bytes auf der Diskette verwendet werden. Wenn Sie nur an einem Teil der Datei interessiert sind, ist es wichtig, daß Sie Teile der Datei überspringen können. Sie möchten zum Beispiel einen speziellen Ausdruck oder eine Zeichenkette durch binäre Suche lokalisieren. (Die binäre Suche ist eine schnelle Methode, eine geordnete Liste zu durchsuchen. Sie wählen den Mittelpunkt und vergleichen die Daten an dieser Stelle mit dem Ausdruck, den Sie suchen. Wenn die Daten kleiner als der Ausdruck sind, wiederholen Sie diese Suchmethode oberhalb dieses Punktes; wenn die Daten größer sind, suchen Sie unterhalb.) Falls die Datei in Ordnung ist, kann das die schnellste Methode sein, einen bestimmten Satz in einer großen Datei zu finden. Oder wenn Ihre binäre Datei eine Bibliothek von Maschinensprache-Routinen enthält, möchten Sie vielleicht nur die Routinen in der zweiten Hälfte der Datei laden. Wie man eine binäre Datei zu diesem Zweck verwendet, wird anhand eines Programmbeispiels am Ende dieses Kapitels besprochen.

GEBRAUCH BINÄRER DATEIEN

Bisher haben wir die Arbeitsweise der Befehle beschrieben; jetzt zeigen wir, wie sie in einem BASIC-Programm angewendet werden können. Die folgenden Abschnitte bringen Beispiele, wie BSAVE und BLOAD bei einer gängigen Grafikanwendung benutzt werden. Danach behandeln wir die Vorteile der Programmierung in Maschinensprache und die Verwendung des BRUN-Befehls zur Ausführung einer Datei.

Eine binäre Datei anlegen

Binäre Dateien werden durch den BSAVE-Befehl automatisch angelegt, wenn der Parameter T nicht benutzt wird. Solche Dateien werden stets als Dateien vom Typ BIN abgespeichert. Auch der CREATE-Befehl kann zum Anlegen einer Datei vom Typ BIN verwendet werden. Mit dem

CREATE-Befehl angelegte Dateien sind leer, bis Sie etwas in sie hineinschreiben.

Das folgende Programm legt drei kleine binäre Dateien zum Abspeichern von Bildern aus dem Speicherbereich der hochauflösenden Grafik des Apple an. Wenn Sie die Bilder einmal erzeugt und abgespeichert haben, können Sie sie jederzeit zurückholen.

```
  10 REM BINAERE PRODOSANWEISUNGEN
  11 REM BSAVE DEMO
  20 HOME
  30 D$ = CHR$(4)
 100 GOSUB 1000
 110 GOSUB 2000
 120 GOSUB 3000
 150 TEXT
 199 END
1000 REM ERSTES BILD MALEN UND ABSPEICHERN
1010 HGR: HCOLOR = 3
1020 HPLOT 5,5 TO 274,5 TO 274,136 TO 5,136 TO 5,5
1030 PRINT D$;"BSAVE BINAER1,A8192,E16383"
1999 RETURN
2000 REM ZWEITES BILD MALEN UND ABSPEICHERN
2010 HGR: HCOLOR = 3
2020 HPLOT 10,10 TO 269,10 TO 269,131 TO 10,131 TO
10,10
2025 HPLOT 130,10 TO 130,131
2030 PRINT D$;"BSAVE BINAER2,A8192,E16383"
2999 RETURN
3000 REM DRITTES BILD MALEN UND ABSPEICHERN
3010 HGR: HCOLOR = 3
3020 HPLOT 15,15 TO 264,15 TO 264,126 TO 15,126 TO
15,15
3025 HPLOT 15,60 TO 264,60
3030 PRINT D$;"BSAVE BINAER3,A8192,E16383"
3999 RETURN
```

In jedem Unterprogramm dieses Programms (1000, 2000 und 3000) wird dieselbe Prozedur verfolgt. Zuerst wird eine HGR-Anweisung ausgeführt, um den Bereich der hochauflösenden Grafik zu löschen und Ihren Apple umzuschalten, damit er anstelle des Speicherbereichs für den Text den Speicherbereich für die Grafik auf dem Bildschirm ausgibt. Als zweites wird im Speicherbereich für die hochauflösende Grafik ein Bild

gemalt. Als drittes wird dieses Bild mit der BSAVE-Anweisung vom Speicher auf eine Diskettendatei übertragen. Die Parameter A und E geben hier den Speicherbereich der hochauflösenden Grafik im Apple an. ProDOS speichert die Startadresse im Verzeichnis und verwendet sie als Voreinstellungsadresse beim Laden der Datei.

Eine binäre Datei laden

Nachdem Sie die Daten in einer binären Datei auf der Diskette gespeichert haben, möchten Sie sie wahrscheinlich einmal zurückholen und in einem anderen Programm benutzen. Das geht mit der BLOAD-Anweisung. Das folgende Programm demonstriert eine typische Anwendung der BLOAD-Anweisung, nämlich das Laden von binären Grafikdaten von der Diskette direkt in den Speicherbereich der hochauflösenden Grafik. Für diese Demonstration verwenden wir die Dateien, die wir im vorherigen Beispiel angelegt haben.

```
  10 REM BINAERE PRODOSANWEISUNGEN
  20 REM BLOAD DEMO
  30 D$ = CHR$(4)
 100 GOSUB 1000
 110 GOSUB 2000
 120 GOSUB 3000
 150 TEXT
 199 END
1000 REM LADEN UND AUSGEBEN VON BILD 1
1005 HGR
1010 PRINT D$;"BLOAD BINAER1"
1030 VTAB 22: INPUT "RETURN ZUM FORTSETZEN DRUEK-
KEN";W$
1999 RETURN
2000 REM LADEN UND AUSGEBEN VON BILD 2
2005 HGR
2010 PRINT D$;"BLOAD BINAER2"
2030 VTAB 22: INPUT "RETURN ZUM FORTSETZEN DRUEK-
KEN";W$
2999 RETURN
3000 REM LADEN UND AUSGEBEN VON BILD 3
3005 HGR
3010 PRINT D$;"BLOAD BINAER3"
3030 VTAB 22: INPUT "RETURN ZUM FORTSETZEN DRUEK-
KEN";W$
3999 RETURN
```

Jedes Unterprogramm in diesem Programm hat den gleichen Ablauf. Zuerst wird eine HGR-Anweisung ausgeführt, mit der der Bereich der hochauflösenden Grafik gelöscht und der Apple umgeschaltet wird, damit er anstelle des Speicherbereichs für den Text den Speicherbereich für die Grafik ausgibt. Dann wird mit der BLOAD-Anweisung eine Datei in den Speicher geladen. Da es sich hier um die Dateien handelt, die wir im vorigen Programm aus dem Bereich der hochauflösenden Grafik abgespeichert haben, hat ProDOS die korrekte Startadresse im Verzeichnis festgehalten und verwendet diese als Standardadresse. Wenn Sie eine Datei benutzen wollen, die aus einem anderen Bereich abgespeichert wurde, müssen Sie mit der Option A die Startadresse angeben (zum Beispiel BLOAD EINEDATEI,A8192). Danach positioniert das Programm den Cursor und wartet, bevor es weiterläuft, auf eine Eingabe, damit Sie Zeit haben, sich das Ergebnis anzuschauen.

Beachten Sie, daß Sie nach dem Laden dieser Dateien keinen weiteren Befehl einzugeben brauchen. Ihr Apple teilt dem Monitor laufend jede Änderung des auszugebenden Speicherbereichs mit. Da sich der Apple im Modus der hochauflösenden Grafik befindet, bewirkt jede Änderung des Speicherbereichs für die hochauflösende Grafik eine Änderung der Anzeige auf dem Bildschirm.

Eine binäre Datei ausführen

Der Hauptgrund dafür, daß Programme in Maschinensprache geschrieben werden, liegt in der viel kürzeren Ausführungszeit. Das liegt daran, daß der Computer keine Zeit für das Interpretieren von Anweisungen zu verwenden braucht; er kann sie einfach ausführen. Andererseits dauert es im allgemeinen viel länger, ein Programm in Maschinensprache oder Assembler (das dann in Maschinensprache umgeformt wird) zu schreiben, als wenn man es in einer höheren Sprache wie etwa in BASIC schreibt. Deshalb sind für höhere Programmiersprachen Compiler entwickelt worden, die einen großen Teil der Ausführungsgeschwindigkeit eines in Maschinensprache geschriebenen Programms zurückgewinnen, ohne dabei die Einfachheit der Programmentwicklung in einer höheren Sprache zu verlieren. Ein solcher Compiler übersetzt die BASIC-Anweisungen in Maschinensprache, um die Ausführungsgeschwindigkeit des Programms zu erhöhen, und speichert diesen Code dann auf einer Diskette ab.

Die Zeitersparnis kommt daher, daß das Programm in Maschinensprachecheform ausgeführt wird. Während so ein schnelleres Programm erstellt wird, verlangsamt der Kompiliervorgang die Entwicklungszeit eines Pro-

gramms, da ein Programm nach jeder Änderung neu kompiliert werden muß. Ein solches Programm, das von einem Compiler erzeugt wurde oder beim Programmieren in Assembler oder Maschinensprache entstanden ist, wird auf der Diskette als binäre Datei gespeichert. Da es ein Programm in der Muttersprache der Maschine ist, braucht es bei seiner Ausführung nicht durch einen Interpreter zu laufen. Aus diesem Grund ist ein Maschinensprache-Programm immer schneller als ein BASIC-Programm, das genau das gleiche macht – jede Anweisung in einem BASIC-Programm muß jedesmal neu interpretiert werden, wenn sie ausgeführt wird. Ein kompiliertes BASIC-Programm läuft also schneller, weil es schon in die Sprache übersetzt worden ist, die der Computer direkt verarbeiten kann. (Anmerkung: Einige Programmiersprachen wie zum Beispiel UCSD-Pascal kompilieren vom Quellcode in einen Pseudo- oder P-Code, der kein Maschinencode ist und deshalb während der Ausführung des Programms interpretiert werden muß.)

In all diesen Fällen sind die Anweisungen der resultierenden binären Datei auszuführen. ProDOS stellt zu diesem Zweck den BRUN- und den Strich-Befehl zur Verfügung. Beim Strich-Befehl wird ein Programm automatisch in den Speicherbereich geladen, aus dem es abgespeichert wurde, und mit seiner Ausführung begonnen. Wie oben erwähnt, können Sie beim BRUN-Befehl mit einem Parameter festlegen, wohin das Programm geladen werden soll.

Binäre Dateien und Maschinensprache-Routinen

Wenn Sie viel programmieren, werden Sie es vorteilhaft finden, einige allgemeine Funktionen in Maschinensprache umzuformen und und auf einer Diskette abzuspeichern, damit Sie sie bei allen Ihren BASIC-Programmen benutzen können. Auf diese Art gewinnen Sie viele Vorteile eines Maschinensprache-Programms, ohne auf BASIC als Ihre Hauptprogrammiersprache verzichten zu müssen.

Die Möglichkeit, ausgewählte Speicherbereiche auf die Diskette zu übertragen und mit dem Parameter B den Bereich der Datei kontrollieren zu können, in den Sie die Daten schreiben, bedeutet, daß Sie Bibliotheksdateien aus Programmen in Maschinensprache anlegen können. Eine Bibliotheksdatei ist eine Datei, die aus vielen verschiedenen Einheiten zusammengesetzt ist, von denen alle voneinander unabhängig sind. Wenn Sie festhalten, wo sich jedes Programm innerhalb der Datei befindet, können Sie die binären Befehle dazu benutzen, nur die Teile der Datei zu laden oder zu speichern, die Sie brauchen. Das bedeutet, daß Sie z. B. eine aus 100 allgemeinen Maschinensprache-Routinen zusammenge-

setzte Bibliotheksdatei von einem Programm in Anspruch nehmen können, das nur einige davon benötigt. Ihr BASIC-Programm kann die Programme, die es braucht, selektiv in den Speicher lesen und alle anderen dabei ignorieren. Das ist eine sehr effektive Technik zum Einsparen von Programmierzeit, Rechenzeit und Speicherplatz.

Nehmen wir zum Beispiel an, Sie wissen, daß die längste der 100 oben erwähnten Routinen 50 Bytes lang ist. Dann können Sie Ihre Routinen so aufteilen, daß genau alle 50 Bytes innerhalb der Datei eine neue Routine anfängt, wobei die erste Routine bei Byte 0 beginnt, die zweite bei Byte 50 und so weiter. Wenn Sie dann eine dieser Routinen einlesen wollen, multiplizieren Sie einfach, um die korrekte Startposition zu erhalten ((Nummer der Routine − 1) * 50). Wenn Sie dann die BLOAD-Anweisung mit der so berechneten Startposition im Parameter B benutzen, erhalten Sie das gewünschte Ergebnis. Nehmen wir an, daß die Nummer der Routine in der Variablen R gespeichert werden soll, dann erledigen das die folgenden Anweisungen für Sie:

```
50000 N = (R-1) * 50
50010 PRINT D$;"BLOAD BIB.ROUTINE,A8000,L50,B";N
```

Die Variable D$ ist natürlich unser alter Bekannter CHR$(4). Die Anweisung in Zeile 50010 lädt genau 50 Bytes von der Datei in den Speicherbereich ab Adresse 8000, angefangen bei der in Zeile 50000 berechneten Position innerhalb der Datei.

Diese Methode verschwendet jedesmal Platz in der Datei, weil kürzeren Routinen auch 50 Bytes zugewiesen werden, um einen festen Abstand zwischen den Routinen beizubehalten. Der einzige Weg, diesen Verlust zu verhindern, ist das jederzeitige Nachhalten der genauen Länge der einzelnen Routinen. Wenn Sie das tun wollen, müssen Sie diese Daten entweder für Ihr Programm zur Verfügung halten (vielleicht indem Sie sie in einem Feld speichern), oder Sie müssen in einer Diskettendatei zu finden sein, die von Ihrem Programm gelesen werden kann. Sie können auch die ersten 100 Bytes Ihrer Datei dazu verwenden, diese Daten in binärer Form zu speichern. Ihr Programm müßte dann diesen Index zuerst lesen, um zu wissen, wo es die einzelnen Routinen finden kann. Dann müßte es die Längen aller vorangehenden Routinen zusammenzählen und 100 Bytes für den Index addieren, um die Startposition einer bestimmten Routine zu berechnen. Routine Nummer 43 beginnt zum Beispiel bei dem Wert, der berechnet wird, indem man die Summe der Längen der ersten 42 Routinen bildet und dazu 100 addiert.

Das läßt die Frage offen, wie man die Routinen an die richtige Stelle auf der Diskette schreibt. Sie sollten zu einer Diskettendatei nur am Dateiende Routinen hinzufügen, oder alle Routinen hinter der neu abgespeicherten gehen verloren. Nehmen wir zum Beispiel an, Sie möchten eine neue Version von Routine 10 in Ihre Datei schreiben. Wenn Sie 50 Bytes für jede vorangehende Routine überspringen, wissen Sie, daß Routine 10 bei Byte 450 in der Datei anfängt ((10 – 1) * 50). Wenn Sie jetzt den BSAVE-Befehl mit dem auf 450 gesetzten Parameter B benutzen, wird die Datei bei Byte 499 enden und die Routinen 11 bis 100 sind verloren. Dies ist sicher nicht das, was Sie wollen.

Wenn Sie jedoch mit dem BLOAD-Befehl den Inhalt der Datei in den Speicher laden, können Sie dort die Routinen nach Belieben editieren und das Ergebnis mit dem BSAVE-Befehl auf die Diskette zurückspeichern. Bei dieser Methode laufen Sie keine Gefahr, Ihr Programm zu verlieren, und Sie haben die gesamte Datei im Speicher, wo Sie sie leichter bearbeiten können. Wenn Sie das im BASIC-Modus machen, können Sie die BASIC-Befehle PEEK und POKE zum Editieren benutzen. Sie können auch den Monitor (in Kapitel 11 beschrieben) des Apple dazu benutzen, den Inhalt des Speichers nach Belieben zu ändern.

Wenn Sie mit dieser Technik Maschinensprache-Programme speichern, vergewissern Sie sich, daß Sie festgehalten haben, was die einzelnen Routinen machen und welche Daten für sie benötigt werden. ProDOS bietet hier keine Hilfmittel an. Sie müssen diese Angaben entweder in einer Diskettendatei oder einfach auf einem Stück Papier aufzeichnen. Wenn diese Aufzeichnung verlorengeht, können Sie sie nur wiedergewinnen, indem Sie Programmzeilen durchlesen. Bei einem Maschinencode ist das aber eine sehr schwierige und unerfreuliche Angelegenheit.

Kapitel 9

Grafik und Ton unter ProDOS

Ihr Apple II bietet Ihnen hinsichtlich seiner Grafik- und Tonfähigkeiten viele Möglichkeiten an, von denen Sie im BASIC-Modus mit den Befehlen GR, HGR und HGR2 Gebrauch machen können. Mit einer geeigneten Abfolge von PEEK- und POKE-Befehlen zu den entsprechenden Speicherstellen können Sie noch genauer vorgehen. Da ProDOS viel Speicherplatz benötigt, ist der richtige Umgang mit den Grafikbereichen unter diesem System besonders wichtig. ProDOS ist ein größeres Betriebssystem als das ältere DOS 3.3 und läßt für den Programmierer weniger Speicherplatz übrig. Als Ausgleich dafür dürfen Applesoft-Programme unter ProDOS den Speicherbereich der hochauflösenden Grafik mitbenutzen, falls das erforderlich ist. In diesem Kapitel behandeln wir, wie man beim Programmieren unter ProDOS aus den Grafik- und Toneigenschaften den richtigen Nutzen ziehen kann.

GRAFIK UND SPEICHER

Die Text- und Grafikausgaben Ihres Apple II werden direkt in bestimmte Speicherbereiche abgebildet. Ihr Apple verwendet drei verschiedene Ausgabemodi: Text, niedrigauflösende Grafik und hochauflösende Grafik. Die Bereiche, die dabei auf dem Bildschirm ausgegeben werden, werden als Seiten im Speicher bezeichnet; jeder Ausgabemodus hat zwei getrennte Seiten für die Ausgabe. Text und niedrigauflösende Grafik teilen sich den gleichen Speicherbereich, während die Ausgabe für die hochauflösende Grafik einen seperaten Speicherbereich benutzt.

Bevor Ihr Apple Daten auf dem Bildschirm ausgibt, schaut er an bestimmten Speicherstellen nach, in welchem Ausgabemodus er sich befindet und welche Seite er ausgeben soll. Bei der Benutzung der BASIC-Befehle TEXT, GR, HGR oder HGR2 werden diese Speicher-

stellen automatisch geändert. Zusätzlich können Sie die Werte in diesen Speicherstellen mit dem POKE-Befehl ändern, wodurch Sie eine noch bessere Kontrolle über die Verwendung des Speicherbereichs und des Ausgabemodus durch Ihren Apple haben.

Während des normalen Betriebs benutzt der Apple den Modus der Textausgabe. Sie können sich aber auch Daten im Modus der niedrig- oder hochauflösenden Grafik ausgeben lassen. In jedem Grafikmodus kontrolliert der Apple die Ausgabe, indem er für einen bestimmten Bildschirmpunkt den Wert in einer bestimmten Speicherstelle nachschaut und entsprechend die Anzeige an diesem einen Punkt ein- oder ausschaltet. Im Modus der niedrigauflösenden Grafik besteht die Anzeige aus 40 Spalten und 48 Zeilen; bei der hochauflösenden Grafik sind es 280 Spalten und 192 Zeilen.

Der Bereich der hochauflösenden Grafik

Die beiden Seiten der hochauflösenden Grafik belegen im Speicher den Bereich von $2000 bis $5FFF (8192–24575). Die erste Hälfte dieses Bereichs stellt Seite 1 und die zweite Hälfte stellt Seite 2 der hochauflösenden Grafik dar. Diese beiden Bereiche sind vom Speicherbereich für die Textausgabe und die niedrigauflösende Grafik unabhängig. Nach dem HGR-Befehl verwendet der Apple Seite 1 als Ausgabebereich, nach dem HGR2-Befehl benutzt er Seite 2.

Der Bereich für den Text und die niedrigauflösende Grafik

Die beiden Seiten für Text und niedrigauflösende Grafik belegen denselben Speicherbereich. Der Unterschied zwischen den beiden Modi besteht in der unterschiedlichen Interpretation der Daten dieses Bereichs. Die gleichen Daten an derselben Speicherstelle führen zu verschiedenen Ausgaben auf dem Bildschirm.

Im Textmodus wird der Wert eines einzelnen Bytes zur Bestimmung des auszugebenden ASCII-Zeichens verwendet. Ein Byte mit dem Wert 86 ($56 oder 01010110 in binärer Schreibweise) veranlaßt zum Beispiel die Ausgabe des Zeichens V. Im Grafikmodus andererseits repräsentiert jedes einzelne Bit eines jeden Bytes einen bestimmten Punkt auf dem Bildschirm. Ein Bit kann die beiden Werte 1 oder 0 haben. Im Grafikmodus wird durch diese beiden Werte bestimmt, ob ein bestimmter Bildschirmpunkt beleuchtet oder ausgeschaltet werden soll. Ein Byte mit dem Wert 86 ($56 oder binär 01010110) repräsentiert zum Beispiel vier beleuchtete und vier ausgeschaltete Bildpunkte.

Die erste Seite des Bereichs für Text und niedrigauflösende Grafik erstreckt sich von $0400 bis $07FF (1024–2047). Seite 2 belegt den Bereich von $0800 bis $0BFF (2048–3071). Da dieser Speicherbereich von beiden Modi benutzt wird, können sich die Bildschirmausgaben beim Umschalten von einem Modus in den anderen gegenseitig beeinträchtigen.

Seite 2 des Bereichs für Text und niedrigauflösende Grafik wird selten benutzt, da sie durch keinen BASIC-Befehl direkt angesprochen werden kann; man kann nur mit POKE-Befehlen auf sie zugreifen.

Der POKE-Befehl und die Ausgabemodi

Mit dem POKE-Befehl können Sie eine Reihe von Umschaltungen im Speicher Ihres Apple vornehmen, um ihm zu sagen, welchen Bereich er ausgeben soll. Die Speicherstellen für diese Umschaltungen sind in Tabelle 9.1 aufgelistet. Diese Befehle werden in den folgenden Abschnitten genauer besprochen, zusammen mit den Ausgabemodi, die sie betreffen.

Stelle	Wirkung
–16297	hochauflösende Grafik
–16298	niedrigauflösende Grafik
–16299	Seite 2
–16300	Seite 1
–16301	Grafikausgabe mit Textfenster
–16302	Ganzschirmgrafik (kein Textfenster)
–16304	nur Text
–16305	Grafik (von Text)

Tabelle 9.1: POKE-Stellen, die die Ausgabebereiche betreffen

NIEDRIGAUFLÖSENDE GRAFIK

Die BASIC-Anweisung GR schaltet die Monitoranzeige vom Textmodus in den Modus der niedrigauflösenden Grafik um. Bei der Ausführung dieser Anweisung wird der Bildschirm gelöscht und die Farbe auf Schwarz gesetzt. Unten am Bildschirm bleibt ein Fenster aus vier Zeilen für den Textbereich offen. Der Computer bleibt bis zur Ausführung eines TEXT-Befehls im Modus der niedrigauflösenden Grafik.

Nach dem TEXT-Befehl wird der gesamte Bildschirm wieder ganz normal zur Textausgabe benutzt. Bei der Ausführung dieses Befehls wird nicht der Bildschirm gelöscht, sondern versucht, die übriggebliebene Grafik als Text zu interpretieren. Das bedeutet, daß Sie möglicherweise eine seltsame Ansammlung von Zeichen auf dem Bildschirm sehen werden. Aus diesem Grund ist es sinnvoll, dem TEXT-Befehl eine HOME-Anweisung folgen zu lassen.

Sie können vom Textmodus aus auch mit den beiden Befehlen

```
POKE -16304,0
POKE -16298,0
```

in den Modus der niedrigauflösenden Grafik umschalten. Der erste Befehl setzt den Ausgabemodus auf Grafik. Der zweite Befehl definiert den Grafikmodus als niedrigauflösend.

Mit der POKE-Anweisung können Sie auch das Textfenster schließen und damit auch die vier Textzeilen am unteren Rand des Bildschirms durch Grafik ersetzen. Die Anweisung

```
POKE -16302
```

läßt das Textfenster verschwinden. Mit dem GR-Befehl können Sie das Textfenster wieder öffnen. Dabei wird allerdings auch der Bildschirm gelöscht und die Farbe auf Schwarz gesetzt. Wenn Sie den Bildschirm nicht löschen wollen, geben Sie einfach den Befehl

```
POKE -16301,0
```

ein. Dann öffnet sich das Textfenster wieder, ohne daß die oberen 40 Grafikzeilen beeinflußt werden. Das folgende Programm demonstriert die Anwendung des POKE-Befehls im Zusammenhang mit dem Textfenster der niedrigauflösenden Grafik:

```
 10 REM NIEDRIGAUFLOESENDE GRAFIK
 11 REM POKE UND DAS TEXTFENSTER
 20 GR: COLOR = 2
 30 HLIN 10,30 AT 5: HLIN 10,30 AT 16
 40 VLIN 6,15 AT 10: VLIN 6,15 AT 30
 50 INPUT "RETURN DRUECKEN ZUM ENTFERNEN DES FEN-
STERS";A$
 60 HOME: POKE -16302,0
 70 COLOR = 0
 72 FOR A = 41 TO 47: HLIN 0,39 AT A: NEXT A
```

```
 74 COLOR = 2
 80 FOR A = 2 TO 38 STEP 2
 82 PLOT A,42 84 NEXT A
 90 FOR A = 1 TO 5000: NEXT A
 95 POKE -16301,0: HOME
100 INPUT "MIT RETURN KOMMEN SIE ZURUECK IN DEN
TEXTMODUS";A$
110 TEXT: HOME
```

Zu Beginn des Programms wird der Bildschirm gelöscht und der Grafikmodus eingeschaltet. In den Zeilen 30 und 40 wird ein Rechteck auf den Bildschirm gemalt. In Zeile 50 wartet das Programm auf die Eingabe eines Return-Zeichens, bevor es weitermacht. Dann wird der Bildschirm gelöscht und das Textfenster mit der POKE-Anweisung in Zeile 60 geschlossen. In den Zeilen 70 bis 84 wird in den Bereich, in dem vorher das Textfenster war, ein Bild gemalt. Die Verzögerungsschleife in Zeile 90 erlaubt Ihnen, sich das Ergebnis anzuschauen, bevor das Fenster wieder eingesetzt wird. Schließlich wartet das Programm noch einmal auf die Eingabe eines Return-Zeichens, bevor es in den Textmodus zurückkehrt.

Es gibt zwei POKE-Befehle, die die Seite des Speichers betreffen, die auf dem Bildschirm ausgegeben wird. Der Befehl

```
POKE -16299,0
```

schaltet die Ausgabe auf Seite 2 des Modus um, in dem Sie sich gerade befinden. Der Befehl

```
POKE -16300,0
```

schaltet entsprechend um auf Seite 1. Die Seite 2 des Bereichs für Text oder niedrigauflösende Grafik können Sie nur durch eine POKE-Anweisung an die Stelle −16299 ausgeben lassen. Die gleichen Anweisungen gelten auch für den Modus der hochauflösenden Grafik.

HOCHAUFLÖSENDE GRAFIK

Die HGR-Anweisung schaltet auf den Modus der hochauflösenden Grafik um und läßt dabei am unteren Rand des Bildschirms ein Textfenster offen. Dabei wird der Bildschirm gelöscht und Seite 1 des Bereichs für die hochauflösende Grafik ausgegeben. Zum Malen wird die bisher verwendete Farbe beibehalten. Weil der Textmodus und der Modus der hochauflösenden Grafik zwei verschiedene Bereiche des Speichers benutzen, ist

der Textbereich immer noch 24 Zeilen lang, obwohl nur vier Zeilen auf dem Bildschirm zu sehen sind. Der Cursor ist nicht sichtbar. Er ist irgendwo außerhalb der vier Textzeilen plaziert.

Mit der HGR2-Anweisung können Sie in den Modus der hochauflösenden Grafik für den gesamten Bildschirm umschalten. Dabei wird Seite 2 des Bereichs für die hochauflösende Grafik gelöscht und auf dem Bildschirm ausgegeben. Bei diesem Befehl wird weder der Inhalt der Seiten für die niedrigauflösende Grafik und der Text noch Seite 1 des Bereichs für die hochauflösende Grafik beeinflußt. Die unteren vier Zeilen des Bildschirms werden bei HGR2 nicht für ein Textfenster reserviert. Die Farbe zum Malen bleibt, wie sie vor der Ausführung der HGR2-Anweisung war.

Weil bei den Anweisungen HGR und HGR2 verschiedene Bereiche des Speichers adressiert werden, können Sie auf Seite 1 und Seite 2 zwei verschiedene Bilder malen und durch Hin- und Herschalten zwischen diesen beiden Seiten die Illusion der Bewegung erzeugen. Sie können auch zwei ganz verschiedene Bilder in den beiden Speicherbereichen stehen haben, wenn Sie wollen, und dann schnell von einer Seite auf die andere umschalten. Benutzen Sie zum Umschalten die Befehle HGR und HGR2, dann gehen die Bilder, die Sie gemalt haben, verloren. Bei diesen Befehlen wird nämlich zuerst der entsprechende Speicherbereich gelöscht, bevor er auf dem Bildschirm ausgegeben wird.

Aus diesem Grund müssen Sie den POKE-Befehl zum Umschalten auf die andere Seite verwenden, wenn Sie verhindern wollen, daß Ihre Bilder zerstört werden. Befinden Sie sich im Modus der hochauflösenden Grafik und wollen Sie Seite 1 auf dem Bildschirm ausgeben, dann geben Sie den Befehl

```
POKE -16300,0
```

ein. Der Befehl

```
POKE -16299,0
```

schaltet die Ausgabe auf Seite 2 um. Beide Befehle haben keine Wirkung, wenn die dort angegebene Seite bereits auf dem Bildschirm ausgegeben wird. Der Vorteil dieser Befehle besteht darin, daß die relevanten Seiten des Speichers nicht geändert werden. (Die gleichen Befehle werden zum Umschalten zwischen den beiden Seiten für Text oder niedrigauflösende Grafik verwendet.)

Wenn Sie sich noch nicht im Modus der hochauflösenden Grafik befinden und eine Grafikseite auf dem Bildschirm darstellen wollen, können Sie

mit dem POKE-Befehl in den Modus der hochauflösenden Grafik umschalten, ohne daß eine Seite des Speicherbereichs für die hochauflösende Grafik gelöscht wird. Vom Modus der niedrigauflösenden Grafik aus geht das mit dem Befehl

```
POKE -16297,0
```

Wenn Sie im Textmodus sind und direkt in den Modus der hochauflösenden Grafik umschalten wollen, geben Sie die Befehle

```
POKE -16304,0
POKE -16297,0
```

ein. Der Inhalt der Speicherbereiche wird von diesen Befehlen nicht beeinflußt. Wenn Sie vorher mit POKE –16299,0 auf Seite 2 der Ausgabe für den Text oder die niedrigauflösende Grafik umgeschaltet hatten, werden Sie sich nach diesen beiden Befehlen auf Seite 2 der hochauflösenden Grafik befinden. Andernfalls sind Sie auf Seite 1.

Befinden Sie sich in irgendeinem Grafikmodus (niedrig- oder hochauflösend), dann kommen Sie mit dem Befehl

```
POKE -16303,0
```

in den Textmodus. Welche Seite des Textbereichs ausgegeben wird, hängt davon ab, auf welcher Grafikseite Sie vor diesem Befehl gewesen sind.

EINE GRAFIKSEITE ABSPEICHERN

Wie wir in Kapitel 8 erwähnt haben, können Sie mit der ProDOS- Anweisung BSAVE den Inhalt einer beliebigen Grafikseite auf eine Diskettendatei übertragen. Alles, was Sie dabei wissen müssen, sind Anfangs- und Endadresse der Seite, die Sie abspeichern wollen. Seite 1 der hochauflösenden Grafik beginnt zum Beispiel bei der Adresse 8192 und endet bei 16383. Mit dem Befehl

```
BSAVE HA1,A8192,E16383
```

können Sie die gesamten 8K des Speicherbereichs von Seite 1 der hochauflösenden Grafik auf eine binäre Datei namens HA1 übertragen. Mit der BLOAD-Anweisung können Sie diese Datei wieder laden und den Inhalt Ihrer Grafikseite wiederherstellen. Da Sie mit der BLOAD-Anweisung eine Datei in einen beliebigen Speicherbereich laden können, können Sie sie genauso einfach in den Bereich ab Adresse 16384 laden und sie über Seite 2 der hochauflösenden Grafik ausgeben lassen.

Erinnern Sie sich, daß Sie sowohl die BLOAD-Anweisung als auch die BSAVE-Anweisung unter ProDOS im Direktmodus verwenden können. Das bedeutet, daß Sie mit dem BSAVE-Befehl den Inhalt der Grafikseiten auch dann noch abspeichern können, wenn Ihr Programm abgebrochen wurde. Geben Sie einfach den BSAVE-Befehl direkt vom BASIC-Prompt aus mit den geeigneten Parametern ein. Wenn Sie nach dem Abbruch des Programms nichts gemacht haben, was den Inhalt einer Grafikseite verändert hat, werden genau die Bilder, die Sie hinterlassen haben, auf die Diskette übertragen.

Mit der BLOAD-Anweisung können Sie auch untersuchen, ob der Inhalt einer Diskettendatei Grafikdaten enthält. Dazu müssen Sie zuerst einen Grafikbefehl ausführen (GR, HGR oder HGR2). Dann laden Sie mit der BLOAD-Anweisung die Datei, die Sie überprüfen wollen, in den Grafikbereich und schauen sich das Ergebnis an. Der Apple interpretiert alle Daten, die Sie in den Ausgabebereich laden, gemäß dem Modus, in dem er sich befindet. Sie werden ein Bild auf dem Schirm sehen, das den Daten in der Datei entspricht. Auch wenn es sich bei der Datei, die Sie überprüfen, um keine Grafikdatei handelt, wird sie auf dem Bildschirm ausgegeben. Sie können dann selbst entscheiden, ob Sie das Bild, das dabei entsteht, als solches bezeichnen können oder nicht.

EINE GRAFIKSEITE SCHÜTZEN

Wenn Sie die Karte mit der Speicheraufteilung Ihres Apple in Anhang J studieren, werden Sie bemerken, daß die Seiten der hochauflösenden Grafik gerade in der Mitte des Bereichs liegen, den der Apple für Applesoft-Programme benutzt. (Das liegt daran, daß ProDOS ein größeres Betriebssystem als DOS 3.3 ist und deshalb weniger Platz für Applesoft-Programme im Speicher übrigläßt.) Wenn Ihr Programm im Speicher wächst (entweder durch Hinzufügen von Programmzeilen oder durch Änderungen am Speicherbedarf der Variablen), kann der Bereich für die hochauflösende Grafik in Anspruch genommen werden. ProDOS schützt diesen Bereich nicht, und er wird von Ihren Programmcode überschrieben, wenn Applesoft den Platz für andere Zwecke braucht.

Die einzige Möglichkeit, diesen Speicherbereich zu schützen, bieten die Anweisungen HIMEM und LOMEM. Sie können zum Beispiel LOMEM auf 24576 setzen, so daß Ihre Applesoft-Programme oberhalb von Seite 2 der hochauflösenden Grafik bleiben. Sie können auch HIMEM auf 8191 setzen, um Ihr ganzes Programm unterhalb von Seite 1 der hochauflösenden Grafik zu halten. Diese Anweisungen schützen ihren Grafikbereich, allerdings auf Kosten der zulässigen Größe Ihrer Programme. Deshalb

kann es besser sein, nur eine Seite der hochauflösenden Grafik zu benutzen und den übrigen Grafikbereich für die Verwendung durch Ihre Programme freizugeben.

Wenn Sie die HIMEM- und LOMEM-Anweisungen zur Kontrolle oder zum Schützen eines Speicherbereichs benutzen wollen, müssen Sie wissen, was sie bedeuten. HIMEM legt die obere Grenze des Speicherbereichs fest, der für Applesoft zur Verfügung steht. LOMEM legt die kleinste Speicherstelle fest, die Applesoft benutzen darf.

Bei der Ausführung eines Programms liegen alle Programmzeilen unterhalb von LOMEM. Alle Variablen werden in dem Bereich zwischen LOMEM und HIMEM gespeichert. Applesoft speichert Variablen nach einem bestimmten System. Ganzzahligen und reellen Variablen wird Speicherplatz in der Reihenfolge ihres Auftretens zugewiesen, angefangen bei LOMEM und aufsteigend in Richtung HIMEM. Variablen für Zeichenketten werden anders angelegt. Der wirkliche Inhalt der Variablen, wird von HIMEM angefangen, abwärts in Richtung LOMEM abgespeichert. Die Zeiger auf diese Zeichenketten werden in gleicher Weise wie ganzzahlige oder reelle Variablen gespeichert (angefangen bei LOMEM und aufwärts in Richtung HIMEM).

Wenn die Variablen für Zeichenketten nach unten in den Bereich der hochauflösenden Grafik hineinwachsen, ersetzen sie die dort gespeicherten Grafikdaten und verändern Ihre Bildschirmausgabe, wenn Sie hochauflösende Grafik benutzen. Das gleiche passiert, wenn die ganzzahligen und reellen Variablen nach oben in den Bereich der hochauflösenden Grafik hineinwachsen.

Wenn Sie Ihr Programm in den Speicher laden, wird LOMEM auf die Stelle gesetzt, die ein Byte oberhalb der letzten Programmzeile liegt. HIMEM wird auf ein Byte unterhalb des Programms BASIC.SYSTEM gesetzt ($BF00). Wenn Sie keine hochauflösende Grafik benutzen wollen, brauchen Sie sich um diese Werte nicht zu kümmern, und der Raum wird vom Programm je nach Bedarf benutzt. Wollen Sie jedoch den Bereich der hochauflösenden Grafik benutzen und ihn schützen, so müssen Sie entweder HIMEM unterhalb oder LOMEM oberhalb dieses Bereichs setzen, damit er sicher bleibt. Da die Größe eines Programms sehr eingeschränkt wird, wenn Sie HIMEM unterhalb des Bereichs für die hochauflösende Grafik setzen, werden Sie normalerweise LOMEM auf einen Wert oberhalb dieses Bereichs setzen.

Der HIMEM-Wert wird durch die ProDOS-Befehle OPEN und CLOSE betroffen. Wenn eine Datei geöffnet wird, reduziert ProDOS den Wert

von HIMEM um 1K (1024 Bytes) und legt einen Puffer der Größe 1K für die Ein- und Ausgabe der Datei an. Dieser Puffer beginnt immer am Anfang einer Speicherseite (eine Seite enthält 256 Bytes) und endet am Ende einer vollständigen Seite. Wenn eine Datei geschlossen wird, kehrt ProDOS diese Prozedur um. Das ist auch dann der Fall, wenn die Datei, die gerade geschlossen wird, nicht die Datei ist, die zuletzt geöffnet wurde. Dann schiebt ProDOS den Inhalt der unteren Puffer entsprechend nach oben.

Wegen dieser ProDOS-Aktionen ändert sich der Wert von HIMEM ständig. Diese Änderungen können Sie minimieren, indem Sie alle Dateien am Anfang Ihres Programms öffnen und sie bis zum Ende Ihres Programms offen lassen. Das ist auch ein weiteres Argument dafür, den HIMEM-Wert nicht unnötig zu verkleinern, weil ProDOS diesen Wert jedesmal reduziert, wenn es einen neuen Puffer anlegen muß. Dann werden nämlich alle Zeichenketten nach unten verschoben und geraten eventuell in den Grafikbereich. Dabei werden die Grafikdaten von den Daten in den Zeichenketten überschrieben. Die Zeichenketten werden als Grafikdaten interpretiert und auf dem Bildschirm ausgegeben.

TÖNE ERZEUGEN

Ihr Apple enthält einen kleinen Lautsprecher, der unter der Tastatur der Maschine liegt. Sie können damit einfache Tonfolgen innerhalb Ihrer Programme erzeugen. Der Lautsprecher ist extrem leicht zu handhaben, weil es nur eine Stelle gibt, über die er angesprochen werden kann. Unglücklicherweise schränkt das auch die Anwendungsmöglichkeiten des Lautsprechers ein. Indem Sie mit dem PEEK-Befehl in die Speicherstelle −16336 hineinschauen, können Sie einen kurzen Ton erzeugen. (Sie können auch den POKE-Befehl dazu benutzen.) Da der Lautsprecher jedesmal kurz ertönt, wenn Sie den PEEK- oder den POKE-Befehl auf diese Stelle anwenden, können Sie die Tonhöhe variieren, indem Sie die Frequenz ändern, mit der Sie die Speicherstelle −16336 ansprechen – je größer die Frequenz, desto höher der Ton. Das folgende Programm demonstriert das:

```
 10 REM TONDEMO
100 HOME
110 VTAB 5: INPUT "VERZOEGERUNG:";D
150 FOR A = 0 TO 100
160 S = PEEK(-16336)
170 FOR B = 1 TO D: NEXT B
190 NEXT A
```

Dieses Programm enthält zwei Schleifen. Die Schleife in Zeile 170 dient als Verzögerungschleife, um die Höhe des Tons zu kontrollieren, den Sie hören, wenn das Programm läuft. Je größer die Verzögerung (die durch den Parameter D im Programm variiert werden kann), desto langsamer kommt der Ton aus dem Lautsprecher.

Mit dem obigen Programm können Sie bestenfalls tiefe Töne erzeugen. Weil Applesoft eine interpretierende Sprache ist, reicht seine Ausführungsgeschwindigkeit nicht aus, um Töne mit hoher Frequenz zu erzeugen. Wenn Sie hohe Töne generieren wollen, müssen Sie ein Programm mit höherer Ausführungsgeschwindigkeit schreiben. Das wäre ein Programm oder ein Unterprogramm in Maschinensprache, das von Ihrem BASIC-Programm aufgerufen werden kann. Dieses Programm muß genau dieselbe Aufgabe erfüllen wie das BASIC-Programm von oben. Der einzige Grund für die größere Ausführungsgeschwindigkeit eines solchen Programms besteht darin, daß es ein Maschinensprache-Programm ist und deshalb nicht jede Zeile bei der Ausführung durch den BASIC-Interpreter interpretiert werden muß.

Wenn Sie schon mit einem Apple II unter DOS 3.3 gearbeitet haben, haben Sie vielleicht festgestellt, daß der Ton des Standardwarnsignals bei ProDOS viel weicher ist als der Piepston bei DOS. Das wird bei ProDOS mit einer kleinen Routine in Maschinensprache erreicht. Sie können Ihre eigenen Routinen in BASIC oder Maschinensprache schreiben und den Lautsprecher nach Belieben benutzen, indem Sie den zeitlichen Abstand ändern, indem Sie die Speicherstelle −16336 ansprechen. Wenn Sie in Assembler schreiben, bewirken die Befehle STA oder LDA mit dieser Speicherstelle, daß der Lautsprecher tönt.

Kapitel 10

ProDOS und das Maschinensprache-Interface

Dieses Kapitel beschreibt das Maschinensprache-Interface (MLI) von ProDOS. Das MLI besteht aus einer Reihe von Routinen, mit denen der Maschinensprache- oder Assembler-Programmierer die von ProDOS angebotenen Möglichkeiten in seinen eigenen Programmen nutzen kann. Die ProDOS-Befehle, die vom BASIC aus oder über das Menü der Dienstprogramme verfügbar sind, benutzen auch diese Routinen. Viele der in diesem Kapitel beschriebenen Funktionen werden Sie schon aus unseren früheren Ausführungen in diesem Buch kennen; haben Sie also keine Angst, wenn sie auf den ersten Blick imponierend und technisch erscheinen. Auch wenn Sie nicht vorhaben, das MLI selbst zu benutzen, bekommen Sie doch eine bessere Vorstellung von der Arbeitsweise von ProDOS, wenn Sie dieses Kapitel lesen.

DIE KOMPONENTEN DES MLI

Wir beginnen mit einigen grundlegenden Definitionen. Das Maschinensprache-Interface (MLI) besteht aus einem unabhängigen Satz von Routinen in Maschinensprache, die in erster Linie mit Diskettendateien zu tun haben. Da es vom ProDOS-Programm BASIC.SYSTEM unabhängig ist, kann es dazu benutzt werden, die Basis für andere Programme zu bilden. Das MLI ist aus vier Komponenten zusammengesetzt, die alle zusammenarbeiten:

- Der Befehlsinterpreter
- Der Dateiverwalter
- Die Laufwerksroutinen
- Das Unterbrechungsprogramm

Der Befehlsinterpreter

Der Befehlsinterpreter im MLI macht die Aufrufnummer und die Parameter eines Aufrufs gültig, bringt die System-Global-Seite auf den neuesten Stand und übergibt die Ausführung an die geeignete Routine des Dateiverwalters. Die Parameter macht er gültig, indem er überprüft, ob die richtige Parameteranzahl als erstes Byte in der Parameterliste des Aufrufs angegeben worden ist. Wenn die Parameter des Aufrufs korrekt sind, bringt der Befehlsinterpreter die System-Global-Seite (Kennzeichenbits und Zeiger zum Aufzeichnen der Systemoperationen) auf den neuesten Stand und übergibt die Kontrolle an den Dateiverwalter. Die Änderungen an der System-Global-Seite werden von ProDOS-Routinen in Maschinensprache vorgenommen. Diese führen die eigentliche Arbeit des Aufrufs aus. Die geänderten Daten der System-Global-Seite werden intern von den MLI-Routinen benutzt. Wir werden sie hier nicht besprechen.

Der Dateiverwalter

Der Dateiverwalter führt alle Aufrufe des MLI aus. Alle Funktionen für den Diskettenzugriff werden durch Aufrufe der Laufwerkroutinen im MLI bewerkstelligt. Der Dateiverwalter merkt sich, welche Laufwerke angesprochen worden sind, führt einfache Speicherverwaltungsfunktionen aus und überwacht den Zustand aller offenen Dateien. Er ruft auch das Unterbrechungsprogramm auf.

Der Dateiverwalter besteht im wesentlichen aus den Hausverwaltungs- und Dateiroutinen des MLI. Die Laufwerkroutinen und das Unterbre-

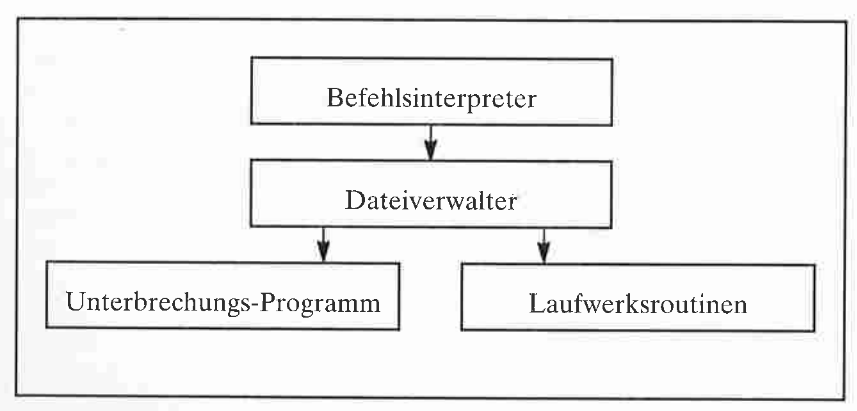


Abb. 10.1: Die Komponenten des MLI

chungsprogramm sind dem Dateiverwalter untergeordnet, weil sie von ihm zur Ausführung ihrer Aufgaben aufgerufen werden. Der Dateiverwalter selbst erhält seine Anweisungen vom Befehlsinterpreter. Diese Struktur ist in Abb. 10.1 illustriert.

Die Laufwerkroutinen

Die Laufwerksroutinen führen die wirklichen Lese- und Schreiboperationen durch, bei denen Daten einer Diskette entnommen oder auf einer Diskette gespeichert werden. Die MLI-Aufrufe (und deshalb auch alle ProDOS-Befehle), die Daten von der Diskette holen oder dort ablegen, verwenden dazu eine der beiden Laufwerkroutinen (READ_BLOCK und WRITE_BLOCK).

Das Unterbrechungsprogramm

Das Unterbrechungsprogramm ermöglicht Ihnen die Installation von bis zu vier Unterbrechungsroutinen. Wenn eine Unterbrechung erzeugt worden ist, ruft das Unterbrechungsprogramm diese Routinen der Reihe nach auf, bis es die Routine findet, die die Unterbrechung ausgelöst hat. Das Unterbrechungsprogramm unterhält eine Tafel mit Vektoren oder Zeigern auf die Eingangsstelle jeder Unterbrechungsroutine. Die Positionen in dieser Tabelle entsprechen der Reihenfolge, in der die Routinen vom MLI installiert worden sind.

PARAMETERLISTEN

Bei MLI-Aufrufen müssen Daten angegeben werden. Diese Daten werden über eine Liste von Parametern übergeben, die entweder die Daten selbst, einen Zeiger auf die Daten oder Speicherplatz für vom Aufruf zurückzugebende Daten enthalten.

Die MLI-Aufrufe erforden einen zwei Bytes langen Zeiger auf eine Parameterliste. Das erste Element dieser Parameterliste ist immer die Parameteranzahl des Aufrufs. Das ist die Zahl, mit der der Befehlsinterpreter überprüft, ob der Aufruf korrekt ist. Ein CREATE-Aufruf hat zum Beispiel sieben Parameter. Wenn Sie bei diesem Aufruf eine andere Anzahl von Parametern angeben, kann das MLI ihn nicht ausführen und bringt den Fehlercode $04 (der falsche Parameteranzahl bedeutet) in den Akkumulator.

Die Parameteranzahl selbst zählt nicht zu den Parametern des Aufrufs. Bei dem CREATE-Aufruf von oben befanden sich in Wirklichkeit acht

Parameter in der Parameterliste: die Parameteranzahl und sieben weitere. Um das deutlich zu machen, hat die Parameteranzahl immer die Nummer 0 in der Parameterliste eines Aufrufs.

Es gibt drei verschiedene Typen von Variablen in einer Parameterliste: Werte, Ergebnisse und Zeiger. Diese Variablen werden dazu benutzt, die MLI-Routine mit Daten zu versorgen und Ergebnisse vom MLI zu übernehmen, wenn der Aufruf beendet ist.

Werte sind ein oder mehrere Bytes lang und werden dazu verwendet, Daten an den Dateiverwalter zu übergeben. Die Daten in diesen Wertparametern helfen dem MLI zu bestimmen, welche Aktion es als Antwort auf Ihren Aufruf ausführen soll. Ein Wert, der an den CREATE-Aufruf übergeben wird, besteht aus einem einzelnes Byte, das den Typ der Datei darstellt, die Sie von ProDOS anlegen lassen wollen.

Ergebnisse sind ein oder mehrere Bytes lang und werden vom Dateiverwalter dazu verwendet, Werte nach dem erfolgreichen Abschluß eines Aufrufs zurückzugeben. Ihr Programm kann diese Ergebnisse lesen und für seine Zwecke benutzen. Der GET_FILE_INFO-Aufruf liefert zum Beispiel einen Ergebniswert aus einem Byte, der Ihnen den Typ der Datei mitteilt, die Sie gerade lesen.

Zeiger sind zwei Bytes lang und zeigen auf eine Speicheradresse. Das erste Byte ist immer der niedrigerwertige Teil der Adresse; das zweite ist das höherwertige Byte. Das bedeutet, daß ein Mensch diese beiden Bytes vertauschen muß, um die korrekte Adresse auszurechnen. (Die Adresse können Sie berechnen, indem Sie den Inhalt des zweiten Bytes mit 256 multiplizieren und das Ergebnis zum Inhalt des ersten Bytes hinzuaddieren.) Zeiger werden verwendet, um dem Dateiverwalter mitzuteilen, wo Daten gespeichert sind oder wo noch Speicherplatz frei ist, in dem die Ergebnisdaten des Aufrufs abgelegt werden können.

PARAMETERTYPEN DES MLI

Die MLI-Aufrufe haben nicht alle die gleichen Parameter. Die Daten, die die einzelnen Routinen für ihre Arbeit benötigen, sind verschieden, und die unterschiedlichen Parameterlisten spiegeln das wieder. In diesem Abschnitt werden diese Parameter kurz anhand ihrer Namen beschrieben, damit eine Grundlage für die Besprechung der MLI-Aufrufe selbst vorhanden ist.

Wie schon vorher erwähnt, besteht die Parameteranzahl aus einem einzelnen Byte, in dem die Anzahl der Parameter des Aufrufs gespeichert

wird. Die Parameteranzahl selbst wird dabei nicht mitgezählt. Bei einem Aufruf mit fünf Parametern wird zum Beispiel die Parameteranzahl auf 5 gesetzt, und fünf Parameter folgen der Parameteranzahl in der Liste.

Ein Pfadnameparameter ist ein zwei Bytes langer Zeiger auf die Adresse einer ASCII-Zeichenkette im Speicher. Das erste Byte der Zeichenkette muß eine Binärzahl enthalten, die gleich der Länge der Zeichenkette ist. Die Zeichenkette selbst darf nicht mehr als 64 Zeichen enthalten. Wenn das erste Zeichen der Zeichenkette kein / ist, wird das momentane ProDOS-Präfix vor die Zeichenkette zur Bildung eines Pfadnamens gesetzt. Der Parameter könnte zum Beispiel die Adresse $3000 enthalten. Ab Adresse $3001 könnte dann die Zeichenkette /PRODOS/VERZEICHNIS stehen.

Der Zugriffsparameter besteht aus einem einzelnen Byte, das bestimmt, wie auf eine Datei zugegriffen werden kann. Individuelle Bits in diesem Byte dienen als Kennzeichenbits, wobei eine 1 an einer jeden Stelle bedeutet, daß ein bestimmter Zugriff erlaubt ist, und eine 0, daß er nicht erlaubt ist. Das Byte ist folgendermaßen aufgebaut:

Bit	**Zugriff**
0	Lesen möglich
1	Schreiben möglich
2	reserviert
3	reserviert
4	reserviert
5	Sicherungskopie erforderlich
6	Umbenennen möglich
7	Löschen möglich

Die Bits 1, 6 und 7 sind auf Null gesetzt, wenn eine Datei schreibgeschützt ist, sonst sind sie auf Eins gesetzt. Die Bits 2 bis 4 sind für zukünftige Erweiterungen von ProDOS reserviert und sollten immer auf Null gesetzt sein. Bit 5 ist auf Eins gesetzt, wenn Sie nicht das Zugriffsbyte eines Dateieintrags mit einem Programm geändert haben, das keinen MLI-Aufruf verwendet. ProDOS setzt das Bit immer auf Eins nach einem CREATE-, RENAME- oder SET_FILE_INFO-Aufruf oder wenn eine Datei nach einem WRITE-Aufruf geschlossen wurde. Dadurch soll festgehalten werden, ob eine Datei geändert worden ist.

Der Dateitypparameter beschreibt den Typ der Datei, mit der Sie es zu tun haben. Dieses einzelne Byte muß einen Wert haben, der einem der Typen in Tabelle 10.1 entspricht.

Dateityp	Bedeutung
$00	Datei ohne Typ (SOS und ProDOS)
$01	Datei mit beschädigten Blöcken
$04	ASCII-Textdatei (SOS und ProDOS)
$06	allgemeine binäre Datei (SOS und ProDOS)
$0F	Verzeichnisdatei (SOS und ProDOS
$12–$BF	SOS-reserviert
$C0–$EF	ProDOS-reserviert
$F0	ProDOS hinzugefügte Befehlsdatei
$F1–$F8	ProDOS-anwenderdefinierte Dateien 1–8
$F9	ProDOS-reserviert
$FA	Integer-BASIC-Programmdatei
$FB	Integer-BASIC-Variablendatei
$FC	Applesoft-Programmdatei
$FD	Applesoft-Variablendatei
$FE	verlegbare Programmdatei (EDASM)
$FF	ProDOS-Systemdatei

Tabelle 10.1: ProDOS-Dateitypen

Der Parameter vom Aux-Typ ist zwei Bytes lang und wird dazu verwendet, die Satzlänge bei Textdateien und die Ladeadresse im Speicher bei binären Dateien festzuhalten. Das Programm BASIC.SYSTEM speichert diese Angaben im Inhaltsverzeichnis, wenn es Dateien behandelt; jedes Programm, das Sie in Assembler oder Maschinensprache schreiben, sollte das gleiche tun, um die Konsistenz mit ProDOS zu erhalten.

Der Parameter für den Speichertyp besteht aus einem Byte und gibt an, wie eine Datei auf der Diskette gespeichert ist. Der Wert $01 in diesem Byte steht für eine Standarddatei; der Wert $D0 repräsentiert eine gebundene Verzeichnisdatei. Alle anderen Werte sind für zukünftige Verwendungen von ProDOS reserviert.

Der Parameter für das Herstellungssdatum benutzt zwei Bytes, um das Datum aufzuzeichnen, an dem eine Datei angelegt worden ist. Der Tag ist in den Bits 0 bis 4 des ersten Bytes enthalten. Der Monat ist in den Bits 5 bis 7 des ersten Bytes und Bit 0 des zweiten Bytes gespeichert. Das Jahr steht in den Bits 1 bis 7 des zweiten Bytes. Ein Schema dazu können Sie in Abb. 10.2 sehen.

Der Parameter für die Herstellungszeit benutzt zwei Bytes, um die Uhrzeit zu speichern, zu der eine Datei angelegt worden ist. Die Minute ist in den Bits 0 bis 5 des ersten Bytes und die Stunde in den Bits 0 bis 4 des zwei-

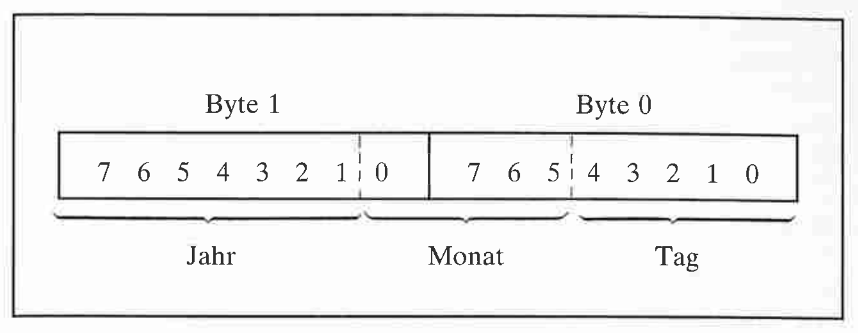


Abb. 10.2: Speicherschema des Parameters für das Herstellungsdatum

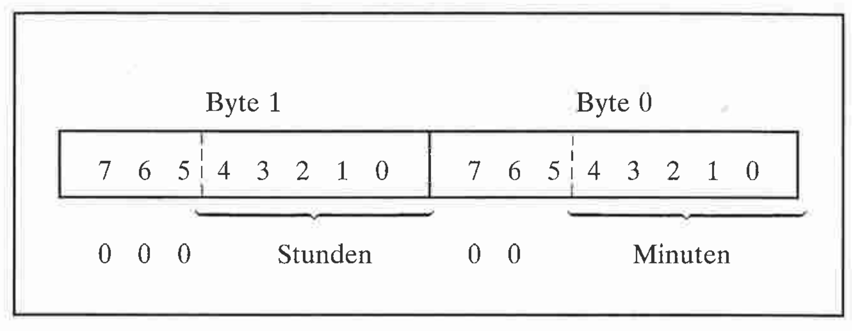


Abb. 10.3: Speicherschema des Parameters für die Uhrzeit der Herstellung

ten Bytes enthalten. Abb. 10.3 illustriert das Speicherschema. Die Bits 6 und 7 des ersten Bytes und 5 bis 7 des zweiten Bytes bleiben immer gleich Null.

Der Parameter für den neuen Pfadnamen wird verwendet, um auf einen zweiten Pfadnamen zu zeigen. Er gehorcht den gleichen Regeln wie der oben beschriebene Parameter für den Pfadnamen, außer daß er auf eine andere Speicherstelle zeigen muß, an der ein zweiter Pfadname abgespeichert worden ist.

Der Mod-Datum-Parameter wird dazu verwendet, das Datum zu speichern, an dem eine Datei zuletzt geändert worden ist. Er hat das gleiche Format wie der Parameter für das Herstellungsdatum.

Der Mod-Zeit-Parameter wird dazu verwendet, die Uhrzeit zu speichern, zu der eine Datei zuletzt geändert worden ist. Die Daten in diesem Byte haben das gleiche Format wie die des Parameters für die Herstellungszeit.

Der Parameter für die belegten Blöcke ist zwei Bytes lang und speichert die Anzahl der von einer Datei belegten Blöcke. Diese Angaben sind im ProDOS-Verzeichnis gespeichert. Wenn Sie sich mit diesem Parameter Angaben über das Stammverzeichnis verschaffen wollen, gibt er die auf der ganzen Diskette belegten Blöcke an und nicht den Platz, der durch das Stammverzeichnis selbst belegt wird.

Der Parameter für die Nummer der Einheit besteht aus einem einzelnen Byte, in dem die Steckplatz- und Laufwerknummern eines Gerätes angegeben sind. In Bit 7 steht die Laufwerknummer (0 bedeutet Laufwerk 1, und 1 bedeutet Laufwerk 2). In den Bits 4 bis 6 ist die Steckplatznummer angegeben. Die Bits 0 bis 3 bleiben unbenutzt.

Der Parameter für den Datenpuffer besteht aus einem zwei Bytes langen Zeiger auf den Anfang des Puffers, der zum Speichern von Angaben über eine Diskette verwendet wird. Die minimale Größe dieses Puffers hängt vom entsprechenden MLI-Aufruf und manchmal auch von den Werten der anderen Parameter des Aufrufs ab. Wir werden das bei jedem einzelnen Aufruf, der diesen Parameter benutzt, besprechen.

Der Parameter für den I/O-Puffer besteht aus einem zwei Bytes langen Zeiger auf einen 1024 Bytes großen Puffer, der für die Ein- und Ausgabeoperationen einer Datei verwendet wird. Dieser Puffer muß am Anfang einer neuen Seite (einem Vielfachen von $0100) im Speicher beginnen und darf nicht aus einem Speicherbereich bestehen, der schon vom System benutzt wird. Bei einer Standarddatei enthalten die ersten 512 Bytes des Puffers den Inhalt des Datenblocks, auf den gerade zugegriffen wird, und die zweiten 512 Bytes den Indexblock (wenn man es mit einfach oder zweifach indizierten Dateien zu tun hat). Bei einer Verzeichnisdatei werden nur die ersten 512 Bytes für den Inhalt des Blocks benutzt, auf den gerade zugegriffen wird; der übrige Teil des Puffers hat keine Verwendung.

Der Parameter für die Referenznummer besteht aus einem Ergebniswert von einem Byte, der für eine Datei zurückgegeben wird, wenn sie mit dem OPEN-Aufruf geöffnet worden ist. Alle folgenden Aufrufe, die diese offene Datei betreffen, müssen diese Zahl als Parameter enthalten. Da nicht mehr als acht Dateien gleichzeitig offen sein dürfen, muß die Zahl zwischen 0 und 7 einschließlich liegen.

Der Maskenparameter besteht aus einem einzelnen Byte zur Bestimmung, welche Bits in einem Zeichen getestet werden sollen. Für eine detaillierte Besprechung dieses Parameters verweisen wir auf den später in diesem Kapitel behandelten NEWLINE-Aufruf.

Der Parameter für das Zeilenwechselzeichen besteht aus einem Byte mit dem Zeichen, das eine Leseoperation beendet. Dieser Parameter wird ebenfalls im Abschnitt über den NEWLINE-Aufruf in diesem Kapitel ausführlicher besprochen.

Der Parameter für die Anforderungsanzahl ist ein Wert aus zwei Bytes, in dem die größtmögliche Anzahl von Bytes angegeben ist, die mit dem Aufruf übertragen werden kann. Diese Zahl darf nicht größer als die Anzahl der Bytes zwischen dem Anfang des Datenpuffers und dem Beginn der nächsten Seite des Speichers sein, die durch die System-Bitabbildung als belegt markiert worden ist.

Der Parameter für die Übertragungsanzahl ist ein Wert aus zwei Bytes, in dem die Anzahl der Bytes abgespeichert wird, die durch einen Aufruf wirklich übertragen worden sind. Aus Gründen, die wir später erklären werden, kann dieser Wert bei einem Aufruf vom Parameter für die Anforderungsanzahl verschieden sein.

Der Positionsparameter ist ein Wert aus drei Bytes, in dem die Stelle innerhalb der Datei angegeben ist, wo die nächste Lese- oder Schreiboperation beginnen wird. Dieser Wert kann nicht größer als der EOF-Wert (Dateiende) der Datei sein. Das erste dieser drei Bytes wird als kleinstes und das letzte als größtes Byte aufgefaßt. Das bedeutet, daß beim Wert $123456 im ersten Byte $56, im zweiten $34 und im dritten $12 gespeichert würde.

Der EOF-Parameter wird dazu verwendet, das Dateiende (End Of File) zu markieren. Dieser Wert ist drei Bytes lang und wird auf die gleiche Weise gespeichert wie der Positionsparameter.

Der Parameter für die Unterbrechungszahl besteht aus einem Byte, das einen Wert von 1 bis 4 enthält. Dieser Wert wird vom Unterbrechungsprogramm bei der Ausführung des ALLOC_INTERRUPT-Aufrufs zugewiesen und beim Entfernen einer Unterbrechungsroutine mit dem DEALLOC_INTERRUPT-Aufruf verwendet.

Der Parameter für den Unterbrechungscode besteht aus einem zwei Bytes langen Zeiger auf den Eintrittspunkt in die Routine, die beim Auftreten einer Unterbrechung vom System aufzurufen ist.

Der Parameter für die Blocknummer besteht aus einem zwei Bytes langen Zeiger, der angibt, welcher Block zur oder von der Diskette übertragen werden soll. Eine normale formatierte Diskette enthält die Blöcke $00 bis $117. Das sind die logischen Blocknummern der Diskette. Disketten sind jedoch in Spuren und Sektoren unterteilt. ProDOS konvertiert logische

Blöcke in Spur- und Sektorpositionen, während es die Diskette liest. Alle Umwandlungen von logischen Blöcken zu physikalischen Spuren und Sektoren werden von den Laufwerkroutinen vorgenommen.

AUFRUFE ANS MLI

Ein MLI-Aufruf besteht aus drei Elementen. Das erste ist der Assembler-Befehl JSR (Jump To Subroutine), mit dem das Programm den Aufruf auslöst. Die beiden anderen Elemente bestehen aus Daten: Das zweite Element besteht aus einem Byte und enthält die Nummer des Aufrufs, der ausgeführt werden soll. Das dritte Element ist drei Bytes lang und enthält einen Zeiger auf die Anfangsadresse der Parameterliste des Aufrufs.

Das MLI überprüft, ob eine gültige Aufrufnummer an es übergeben wurde. Die gültigen Aufrufnummern sind in Tabelle 10.2 aufgelistet.

Aufruf	deutsche Bedeutung	Aufrufnummer
CREATE	anlegen	$C0
DESTROY	zerstören	$C1
RENAME	umbenennen	$C2
SET_FILE_INFO	Dateiinfo setzen	$C3
GET_FILE_INFO	Dateiinfo lesen	$C4
ON_LINE	angeschlossen	$C5
SET PREFIX	Präfix setzen	$C6
GET PREFIX	Präfix lesen	$C7
OPEN	öffnen	$C8
NEWLINE	neue Zeile	$C9
READ	lesen	$CA
WRITE	schreiben	$CB
CLOSE	schließen	$CC
FLUSH	Puffer retten	$CD
SET_MARK	Position setzen	$CE
GET_MARK	Position lesen	$CF
SET_EOF	Dateiende setzen	$D0
GET_EOF	Dateiende lesen	$D1
SET_BUF	Pufferadr. setzen	$D2
GET_BUF	Pufferadr. lesen	$D3
ALLOC_INTERRUPT	Unterbrechung anlegen	$40
DEALLOC_INTERRUPT	Unterbrechung entfernen	$41
READ_BLOCK	Block lesen	$80
WRITE_BLOCK	Block schreiben	$81
GET_TIME	Zeit ablesen	$82

Tabelle 10.2: Die MLI-Aufrufe

Sie müssen einen MLI-Aufruf auf eine bestimmte Art durchführen. Die JSR-Anweisung muß das Programm zur Adresse $BF00 hinführen, wo sich der Eintrittspunkt des MLI befindet. Der Anweisung muß als erstes ein einzelnes Byte mit der Aufrufnummer und als zweites ein zwei Bytes langer Zeiger (mit dem kleineren Byte zuerst) zu der Adresse der Parameterliste folgen.

Wenn der MLI-Aufruf abgeschlossen ist, geht die Ausführung des Programms drei Bytes hinter der JSR-Anweisung weiter. Es ist gute Programmiertechnik, an dieser Stelle einen Fehlertest zu machen. Das MLI gibt beim Auftreten eines Fehlers nämlich keine Fehlermeldung auf dem Bildschirm aus. Sie müssen dann selbst herausfinden, was falsch ist. Der nächste Abschnitt „MLI-Fehler" enthält darüber weitere Informationen.

Es folgt ein Beispiel eines Assembler-Unterprogramms für die Ausführung eines MLI-Aufrufs:

```
AUFRUF  JSR  $BF00   ;den Befehlsinterpreter aufrufen
        DB   CNUMMR  ;MLI-Aufrufnummer
        DW   PARLIST ;zeigt auf Parameterliste
        BNE  FEHLER  ;Fehlertest
        RTS
```

Die erste Anweisung veranlaßt dieses Unterprogramm, einen Sprung zur Unterroutine zu machen, die bei $BF00 anfängt (JSR ist der BASIC-Anweisung GOSUB ähnlich). Darauf folgt eine Anweisung, die ein einzelnes Byte im Speicher, die Variable CNUMMR, als Datenbyte definiert. Dort sollte vor der Ausführung von AUFRUF die Nummer des MLI-Aufrufs abgelegt werden. Die dritte Anweisung definiert ein Wort (zwei Bytes) im Speicher (die Variable PARLIST) als Speicherstelle für Daten. An die Variable PARLIST muß vor der Ausführung von AUFRUF der Zeiger auf die Parameterliste gegeben werden. Die vierte Anweisung führt einen Test zur Fehlerüberprüfung durch und zweigt zu einer Routine für die Fehlerbehandlung ab, wenn ein Fehler bemerkt wird. Das Unterprogramm endet schließlich mit der Rückkehranweisung (RTS, ähnlich der BASIC-Anweisung RETURN).

Vor der Ausführung von AUFRUF muß ein Programm die korrekten Werte in den Variablen CNUMMR und PARLIST gespeichert haben. Andernfalls versucht das MLI einen Aufruf mit den Daten durchzuführen, die sich an diesen Stellen gerade befinden. Das könnten die Daten von einem früheren Aufruf sein, so daß dieser MLI-Aufruf wiederholt würde, oder die Daten könnten die Bedingungen an die Parameter nicht erfüllen, so daß ein Fehler auftreten würde.

Alle MLI-Aufrufe kehren zum Schluß an die Anweisung zurück, die drei Bytes hinter der Adresse der JSR-Anweisung liegt, die den Aufruf initiiert hat. Deshalb haben wir die BNE-Anweisung dorthin gesetzt, damit sie als erste Anweisung hinter dem MLI-Auruf ausgeführt wird. In diesem Falle ist FEHLER eine Marke (ähnlich einer Zeilennummer in BASIC), um eine bestimmte Stelle innerhalb des Programms zu identifizieren. Wenn ein Fehler auftritt, veranlaßt diese Anweisung, daß das Programm zu der durch FEHLER markierten Stelle verzweigt. Jeder von Null verschiedene Wert im Akkumulator zeigt einen Fehler an.

Der GET_TIME-Aufruf hat keine Parameter und mag als Ausnahme zu dem geforderten Format eines MLI-Aufrufs erscheinen. Während der Inhalt des Zeigers auf die Parameterliste bei diesem Aufruf bedeutungslos ist (mit anderen Worten, es ist egal, wo der Zeiger hinzeigt), ist doch der vom Zeiger belegte Platz von Bedeutung. Nach einem MLI-Aufruf geht die Ausführung eines Programms immer drei Bytes hinter der JSR-Anweisung weiter, die den Aufruf ausgelöst hat. Aus diesem Grund müssen Sie sowohl eine Aufrufnummer als auch einen Zeiger auf die Parameterliste angeben, und Sie müssen sich vergewissern, ob die als nächstes ausgeführte Anweisung auch wirklich die ist, die Sie dafür vorgesehen haben. Wenn Sie im Beispiel von oben die DW-Anweisung weglassen, würde die Programmausführung mit einem Teil der Adresse von FEHLER weitergehen. Da dort alles Mögliche stehen kann, wird das Ergebnis unvorhersehbar und mit ziemlicher Sicherheit fatal für Ihr Programm sein.

MLI-FEHLER

Bei einem MLI-Aufruf können zwei verschiedene Ergebnisse auftreten. Entweder ist ProDOS in der Lage, den Aufruf erfolgreich durchzuführen oder der Aufruf mißlingt. Dieses Ergebnis ist offensichtlich für Ihr Programm von Bedeutung. Das MLI übergibt Daten, die dieses Ergebnis kennzeichnen, an die Register Ihres Apple. (Die Register sind bestimmte Speicherstellen zum Aufzeichnen von Statusangaben und für Situationen, in denen ein schneller Datenzugriff erforderlich ist.)

In der folgenden Tabelle ist der Registerzustand Ihres Apple nach der Durchführung eines MLI-Aufrufs aufgelistet. Die mit N, Z, C, D und V markierten Spalten stellen Kennzeichenbits dar; sie können nur die Werte Null oder Eins annehmen und werden dazu verwendet, Statusänderungen anzuzeigen. Das Akkumulatorregister (Acc) und das X-, Y-, und Stapelzeigerregister sind je acht Bits breit. Der Programmzähler (PC) besteht aus 16 Bits und enthält die Adresse der nächsten auszufüh-

renden Anweisung. Nach dem Abschluß eines MLI-Aufrufs zeigt der Programmzähler immer auf die Adresse der Anweisung, die drei Stellen hinter der Anweisung kommt, mit der das MLI aufgerufen wurde. Wir haben dort unsere Anweisung zur Fehlerüberprüfung abgelegt, so daß sie unmittelbar hinter dem MLI-Aufruf ausgeführt wird.

	NZCDVAcc	**PC**	**X Y SP**
erfolgreicher Aufruf:	0 1 0 0 x 0	JSR+3	unverändert
mißlungener Aufruf:	0 0 1 0 x Fehlercode	JSR+3	unverändert

Der Wert des Bedingungscode N wird durch den Wert im Akkumulator bestimmt; er wäre nur bei einem Fehlercode, der größer als $80 ist (den es bis jetzt nicht gibt), gleich 1. Der Fehlercode selbst wird immer in den Akkumulator nach einem nicht erfolgreichen MLI-Aufruf zurückgegeben. Das Bit V ist im Zusammenhang der MLI-Aufrufe undefiniert (das heißt, das MLI übergibt kein Ergebnis an den Bedingungscode, und er kann sowohl auf 0 als auch auf 1 gesetzt sein). Bei einem erfolgreichen Aufruf wird Z auf 1 und auf 0 gesetzt; bei einem Fehler wird Z auf 0 und C auf 1 gesetzt. D ist immer auf 0 gesetzt.

Wie Sie sehen, sind sowohl die Bedingungscodes Z und C als auch der Akkumulator vom Erfolg eines MLI-Aufrufs betroffen. Sie können einen beliebigen von diesen drei Werten überprüfen, um festzustellen, ob ein Fehler aufgetreten ist. Die BNE-Anweisung im Programmsegment von oben überprüft das Bit Z. Wenn das Bit Z auf 0 gesetzt ist und einen Fehler anzeigt, verzweigt das Programm nach FEHLER, der Routine, die den Akkumulator überprüft. Der Wert im Akkumulator teilt Ihnen durch eine Codezahl mit, welcher Fehler aufgetreten ist.

Die Fehlercodes, die das MLI an den Akkumulator übergeben kann, sind in Tabelle 10.3 dargestellt. Sie können in jeder Routine zur Fehlerbehandlung, die Sie schreiben, den Akkumulator zur Bestimmung des aufgetretenen Fehlers ablesen und dann die Aktion durchführen, die Sie gewählt haben. Das Einfachste ist bei der Fehlerbehandlung meistens das Beste. Da Ihnen das MLI über eine Meldung auf dem Bildschirm nicht mitteilt, ob ein Fehler aufgetreten ist, sollte eine solche Routine mindestens eine Meldung wie

```
MLI-AUFRUF MISSLUNGEN ->FEHLERNR.:
```

ausgeben, dahinter den im Akkumulator vorgefundenen Wert.

Mit den Aufrufen READ_BLOCK und WRITE_BLOCK können Sie direkt auf den Inhalt eines logischen Blocks auf einer Diskette zugreifen.

Code	Beschreibung
$00	kein Fehler
$01	falsche Systemaufrufzahl
$04	falsche Parameteranzahl des Systemaufrufs
$25	Unterbrechungstafel voll
$27	I/O-Fehler
$28	kein Gerät angeschlossen
$2B	Diskette schreibgeschützt
$2E	Diskette ausgetauscht
$40	ungültiger Pfadname
$42	maximale Anzahl von Dateien offen
$43	Ungültige Bezugszahl
$44	Verzeichnis nicht gefunden
$45	Datenträger nicht gefunden
$46	Datei nicht gefunden
$47	doppelter Dateiname
$48	Datenträger voll
$49	Stammverzeichnis voll
$4A	Dateiformat nicht kompatibel
$4B	nicht unterstützter Speichertyp
$4C	aufs Dateiende gestoßen
$4D	Position außerhalb des Bereichs
$4E	Dateizugriffsfehler oder Datei geschützt
$50	Datei offen
$51	Verzeichnisstruktur beschädigt
$52	kein ProDOS-Datenträger
$53	ungültiger Parameter des Systemaufrufs
$55	Tabelle des Datenträgerkontrollblocks voll
$56	falsche Pufferadresse
$57	doppelter Datenträger
$5A	Dateistruktur beschädigt

Tabelle 10.3: Die Fehlercodes des MLI

Diese Befehle werden normalerweise bei Dienstprogrammen verwendet und sollten mit Vorsicht benutzt werden. Die Dateistruktur von ProDOS kann beschädigt werden, wenn eine WRITE_BLOCK-Anweisung notwendige Informationen durch falsche Daten ersetzt.

DIE MLI-AUFRUFE

Es gibt drei Kategorien von MLI-Aufrufen: Hausverwaltungsaufrufe, Dateiaufrufe und Systemaufrufe. Hausverwaltungsaufrufe werden im all-

gemeinen dazu verwendet, den Status einer Datei zu ändern, während Dateiaufrufe dazu benutzt werden, Daten von einer oder auf eine Diskette zu übertragen. Systemaufrufe sind alle Aufrufe, die nicht in die ersten beiden Kategorien passen. Eine Nachschlageliste mit allen MLI-Aufrufen und ihren Parametern ist in Anhang G zu finden.

Hausverwaltungsaufrufe

Unter ProDOS bestehen Hausverwaltungsaufrufe aus solchen Operationen, die nicht an offenen Dateien vorgenommen werden können oder nichts mit offenen Dateien zu tun haben. Das schließt Operationen wie Anlegen und Löschen von Dateien ein. Das ProDOS-MLI besitzt folgende acht Hausverwaltungsaufrufe:

CREATE
DESTROY
RENAME
SET_FILE_INFO
GET_FILE_INFO
ON_LINE
SET_PREFIX
GET_PREFIX

CREATE ($C0)

Mit dem CREATE-Aufruf kann entweder eine Standard- oder eine Verzeichnisdatei angelegt werden. Sie können damit keine Stammverzeichnisdatei herstellen, denn diese wird nur beim Formatieren einer Diskette angelegt. Dieser Befehl erzeugt einen Eintrag in dem im Pfadnamen angegebenen Verzeichnis und reserviert für diese Datei auf der Diskette einen Block Speicherplatz.

Der CREATE-Aufruf hat sieben Parameter. Sie sind hier mit ihrer Länge in Bytes und ihrem Typ in der Reihenfolge aufgelistet, in der sie erscheinen müssen.

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 7 sein)
1.	Pfadname	2	Zeiger
2.	Zugriff	1	Wert
3.	Dateityp	1	Wert
4.	Aux-Typ	2	Wert

5.	Speichertyp	1	Wert
6.	Anlegedatum	2	Wert
7.	Anlegeuhrzeit	2	Wert

DESTROY ($C1)

Der DESTROY-Aufruf löscht die angegebene Datei von der Diskette. Der Verzeichniseintrag für diese Datei wird entfernt, und die von der Datei belegten Blöcke werden in der System-Bitabbildung als zu vergeben markiert. ProDOS setzt einige Grenzen bei der Verwendung dieses Befehls. Sie können damit nicht eine Stammverzeichnisdatei oder eine zur Zeit offene Datei löschen. Auch Unterverzeichnisdateien können Sie nicht löschen, es sei denn, sie sind leer. Diese Einschränkungen sind zum Schutz der Diskette vorgesehen und um die Möglichkeit eines katastrophalen Fehlers zu verringern, wie etwa, daß eine Unterverzeichnisdatei, gelöscht wird, die noch weitere Dateien enthält. Schreibgeschützte Dateien können mit dem DESTROY-Aufruf nicht gelöscht werden; zuerst muß der Schreibschutz entfernt werden.

Der DESTROY-Aufruf hat nur einen Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 1 sein)
1.	Pfadname	2	Zeiger

RENAME ($C2)

Der RENAME-Aufruf ändert den Namen einer bestehenden Datei in einen neuen Pfadnamen um. Alter und neuer Pfadname dürfen sich nur im letzten Eintrag unterscheiden. Das Ergebnis dieses Befehls ist die Änderung eines Dateinamens innerhalb einer festen Verzeichnisdatei. Dieser Aufruf erfüllt genau die gleiche Aufgabe wie der Befehl zum Umbenennen von Dateien in den Datei-Dienstprogrammen.

Der RENAME-Aufruf hat zwei Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 2 sein)
1.	Pfadname	2	Zeiger
2.	neuer Pfadname	2	Zeiger

SET_FILE_INFO ($C3)

Der SET_FILE_INFO-Aufruf wird dazu verwendet, die in einem Verzeichnis gespeicherten Angaben über eine bestimmte Datei zu modifizieren. Die Änderungen durch diesen Aufruf werden von einer offenen Datei nicht wahrgenommen, bis sie geschlossen und wieder geöffnet worden ist. Die Parameter im SET_FILE_INFO-Aufruf basieren auf Daten, die man mit dem GET_FILE_INFO-Aufruf zurückbekommen kann. Um hier Fehler zu vermeiden, sollten Sie die Daten zuerst mit dem GET_FILE_INFO-Aufruf einlesen, bevor Sie sie mit dem SET_FILE_INFO-Aufruf modifizieren und ins Verzeichnis zurückübertragen.

Der SET_FILE_INFO-Aufruf hat sieben Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 7 sein)
1.	Pfadname	2	Zeiger
2.	Zugriff	1	Wert
3.	Dateityp	1	Wert
4.	Aux-Typ	2	Wert
5.	Nullfeld	3	
6.	mod. Datum	2	Wert
7.	mod. Uhrzeit	2	Wert

Obwohl Sie mit diesem Aufruf die Angaben im Verzeichnis einer Datei modifizieren können, während die Datei offen ist, merkt ProDOS diese Änderungen erst, nachdem die Datei geschlossen worden ist. Für jede offene Datei ist zum Zeitpunkt der Öffnung ein Datei-Konntrollblock angelegt worden, der nicht geändert wird, bis die Datei geschlossen und wieder geöffnet worden ist. Das geschieht, um Änderungen an einer offenen Datei zu verhindern, die Lese- und Schreiboperationen auf dieser Datei betreffen. Angenommen, Sie haben eine Datei als Textdatei mit der Satzlänge 10 geöffnet. Während die Datei noch offen ist, kann Ihr Programm einen SET_FILE_INFO-Aufruf durchführen und die Satzlänge auf 20 setzen. Solange die Datei geöffnet bleibt, behandelt ProDOS sie als eine Datei mit der Satzlänge 10. Erst wenn Sie sie schließen und wieder öffnen, übernimmt es die neue Satzlänge 20.

Das drei Bytes lange Nullfeld in der Parameterliste ist nur dazu vorgesehen, die Symmetrie zwischen diesem Aufruf und dem GET_FILE_INFO-Aufruf zu wahren. Diese drei Bytes füllen nur leeren Raum aus. Das erleichtert es, beim SET_FILE_INFO-Aufruf die mit dem GET_FILE_INFO-Aufruf eingelesenen Werte zu benutzen.

GET_FILE_INFO ($C4)

Der GET_FILE_INFO-Aufruf liest die Angaben über eine bestimmte Datei im Verzeichnis. Dieser Aufruf kann bei offenen und geschlossenen Dateien verwendet werden. Wenn jedoch eine offene Datei mit dem SET_FILE_INFO-Befehl modifiziert worden ist, wird der mit dem GET_FILE_INFO-Befehl gelesene Zugriffswert erst gültig, wenn die Datei geschlossen und wieder geöffnet worden ist. Das liegt daran, daß der beim Öffnen der Datei angelegte Datei-Kontrollblock (FCB), der oben beschrieben worden ist, vom SET_FILE_INFO-Aufruf unberührt bleibt.

Der GET_FILE_INFO-Aufruf hat folgende 10 Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich $A sein)
1.	Pfadname	2	Zeiger
2.	Zugriff	1	Resultat
3.	Dateityp	1	Resultat
4.	Aux-Typ	2	Resultat
5.	Speichertyp	1	Resultat
6.	belegte Blöcke	2	Resultat
7.	mod. Datum	2	Resultat
7.	mod. Uhrzeit	2	Resultat
8.	Anlegedatum	2	Resultat
9.	Anlegeuhrzeit	2	Resultat

ON_LINE ($C5)

Der ON_LINE-Aufruf gibt die Namen aller ProDOS-Datenträger an, die zur Zeit zur Verfügung stehen. Sie können damit auch den Namen einer Diskette in einem Laufwerk bestimmen, wenn Sie Steckplatz- und Laufwerknummer angeben. Wenn die Nummer einer existierenden Einheit übergeben wird, werden der Diskettenname und Steckplatz- und Laufwerknummer des angegebenen Laufwerks in einem Puffer gespeichert. Für diese Angaben ist ein Platz von nur 16 Bytes erforderlich.

Ist die Einheitsnummer jedoch auf 0 gesetzt worden, sollte der Datenpuffer mindestens 256 Bytes lang sein. Denn wenn 0 als Einheitsnummer übergeben worden ist, werden Datenträgername, Steckplatznummer und Laufwerknummer aller angeschlossener Laufwerke in den Puffer gelesen. Da für jedes Laufwerk 16 Bytes Speicherplatz benötigt werden,

reicht das auch für den „ungünstigsten“ Fall: wenn an alle 8 Steckplätze zwei Laufwerke angeschlossen sind.

Der ON_LINE-Aufruf hat nur zwei Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 2 sein)
1.	Einheitsnummer	1	Wert
2.	Datenpuffer	2	Zeiger

SET_PREFIX ($C6)

Der SET_PREFIX-Befehl setzt das Standardpräfix auf das angegebene Verzeichnis. Dabei gelten alle vorher besprochenen Regeln für einen gültigen Pfadnamen. Das MLI überprüft, ob das angegebene Verzeichnis erreichbar ist, bevor es den Aufruf akzeptiert. Dieser Aufruf hat die gleiche Funktion wie der SET PRODOS PREFIX-Befehl in den Datei-Dienstprogrammen oder der PREFIX-Befehl im BASIC.

Der SET_PREFIX-Aufruf hat nur einen Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 1 sein)
1.	Pfadname	2	Zeiger

GET_PREFIX ($C7)

Der GET_PREFIX-Befehl gibt das gegenwärtige Standardpräfix an. Wenn kein Standardpräfix existiert, wird 0 angegeben. Andernfalls wird das (in Schrägstriche eingeklammerte) Systempräfix in den Datenpuffer, der 128 Bytes lang sein muß, übertragen.

Der GET_PREFIX-Aufruf hat nur einen Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 1 sein)
1.	Datenpuffer	2	Zeiger

Dateiaufrufe

Die ProDOS-Dateiaufrufe haben mit der Übertragung von Daten zu oder von einer Diskettendatei zu tun. Alle Dateien müssen mit dem OPEN-

Aufruf geöffnet worden sein, bevor ein anderer Dateiaufruf auf ihren Inhalt Einfluß nehmen kann.

Das MLI hat folgende zwölf Dateiaufrufe:

OPEN
NEWLINE
READ
WRITE
CLOSE
FLUSH
SET_MARK
GET_MARK
SET_EOF
GET_EOF
SET_BUF
GET_BUF

OPEN ($C8)

Der OPEN-Aufruf bereitet eine Datei so vor, daß sie gelesen oder in sie geschrieben werden kann. Er gibt eine Referenznummer zurück, die der Datei zugeteilt bleibt, solange sie offen ist. Dieser Wert wird von allen anderen Dateiaufrufen, die sich auf diese Datei beziehen, benutzt. Der OPEN-Aufruf legt einen Datei-Kontrollblock (FCB) innerhalb des I/O-Puffers an, um sich die momentanen Eigenschaften der Datei zu merken. Der laufende Positionsanzeiger wird auf 0 gesetzt. Der Datei-Kontrollblock der Datei wird beibehalten, bis die Datei wieder geschlossen wird.

Der I/O-Puffer wird für alle Operationen verwendet, die mit dem Beschreiben oder Lesen einer Diskette zu tun haben. Er ist immer 1024 Bytes (1K) lang. Entfernen Sie eine Diskette nicht aus dem Laufwerk, solange noch eine Datei offen ist – ProDOS überprüft vor dem Lesen oder Schreiben nicht, ob noch diese Diskette da ist. Wenn Sie eine andere Diskette ins Laufwerk gelegt haben, geht ProDOS davon aus, daß sich die Datei noch genau da befindet, wo sie beim Öffnen war. Bei einer Leseoperation werden dann falsche Daten in den Dateipuffer gelesen. Eine Schreiboperation kann noch unangenehmere Folgen haben – wichtige Daten können überschrieben werden und verlorengehen. Werden die Daten in den Bereich eines Verzeichnisses geschrieben, so werden die Dateieinträge dieser Diskette unlesbar, und Sie können auf die entsprechenden Dateien nicht mehr zugreifen.

Sie können zwei Dinge tun, um das zu verhindern: Entfernen Sie keine Disketten aus einem Laufwerk, solange noch Dateien offen sind; lassen Sie Ihr Programm mit einem ON_LINE-Aufruf die Identität einer Diskette überprüfen, bevor Sie auf sie schreiben.

Erinnern Sie sich, daß nicht mehr als acht Dateien unter ProDOS gleichzeitig geöffnet sein dürfen. Jeder Versuch, mehr Dateien zu öffnen, erzeugt die Fehlermeldung, daß die Kontrollblocktafel voll ist.

Wird eine Datei geöffnet, so wird ihr eine Stufenzahl zugewiesen, die auf dem Wert der entsprechenden Stelle der System-Global-Seite basiert. Diese Zahl liegt immer zwischen 0 und 7. Dieser Wert ist wichtig, weil er beim CLOSE-Aufruf verwendet wird.

Der OPEN-Aufruf hat folgende drei Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 3 sein)
1.	Pfadname	2	Zeiger
2.	I/O-Puffer	2	Zeiger
3.	Referenznummer	1	Ergebnis

NEWLINE ($C9)

Mit dem NEWLINE-Aufruf können Sie bei einer beliebigen offenen Datei den Zeilenwechsel-Modus setzen oder rückgängig machen. Wenn er rückgängig gemacht ist, endet eine Leseoperation nur dann, wenn sie auf die Markierung des Dateiendes trifft oder wenn eine angegebene Anzahl von Zeichen eingelesen worden ist. Ist der Zeilenwechsel-Modus gesetzt, wird ein Lesevorgang auch beendet, wenn das Zeichen für den Zeilenwechsel innerhalb einer Datei gelesen wird.

Mit dem Maskenparameter wird bestimmt, welche Bits des eingelesenen Zeichens mit dem Zeichen für den Zeilenwechsel verglichen werden sollen. Nachdem das Zeichen in den Puffer gelesen wurde, wird eine logische AND-Operation (UND-Verknüpfung) mit der Maske durchgeführt und das Ergebnis mit dem Zeilenwechsel-Zeichen verglichen. Wenn diese beiden übereinstimmen, wird der Lesevorgang beendet. Das eingelesene Zeichen wird durch die AND-Operation nicht verändert.

Der Wert $00 im Maskenparameter hat zur Folge, daß die Wirkung des Zeilenwechsel-Modus annulliert wird, da keine Bits verglichen werden. Jeder andere Wert setzt den Zeilenwechsel-Modus. Der Wert $FF

bewirkt, daß alle Bits verglichen werden. Der Wert $7F (der die Maske auf die Binärzahl 01111111 setzen würde) würde bewirken, daß die Bits 0 bis 6 verglichen werden und Bit 7 nicht. Bei dieser Maske würde das ASCII-Zeichen 13 ($0D) oder 00001101 in binärer Schreibweise mit sich selbst und mit dem Zeichen $8D (binär 10001101) übereinstimmen. Bit 7 wird nämlich nicht verglichen, und die Bits 0 bis 6 sind identisch.

Der NEWLINE-Aufruf hat folgende drei Parameter:

	Inhalt	**Länge**	**Typ**
0.	Parameteranzahl	1	Wert (muß gleich 3 sein)
1.	Referenznummer	1	Wert
2.	Maske	1	Wert
3.	Zeilenwechsel-Zeichen	1	Wert

READ ($CA)

Der READ-Aufruf überträgt Daten von der Diskette in den Speicher. Die Anzahl der angeforderten Bytes stimmt immer mit der Anzahl der übertragenen Bytes überein, es sei denn, während des Lesevorgangs wurde das Dateiende oder das Zeilenwechsel-Zeichen bei gesetztem Zeilenwechsel-Modus angetroffen. Wenn eine dieser Situationen auftritt, gibt der Parameter für die Übertragungsanzahl die Zahl der Bytes an, die tatsächlich übertragen worden sind, einschließlich des Bytes für das Zeilenwechsel-Zeichen. Ein Dateiende-Fehler tritt nur dann auf, wenn beim READ-Aufruf Nullzeichen übertragen wurden. (Mit anderen Worten: Das Dateiende war schon vor der Ausführung des Aufrufs erreicht.)

Die READ-Anweisung hat folgende vier Parameter:

	Inhalt	**Länge**	**Typ**
0.	Parameteranzahl	1	Wert (muß gleich 4 sein)
1.	Referenznummer	1	Wert
2.	Datenpuffer	2	Zeiger
3.	Anforderungsanzahl	2	Wert
4.	Übertragungsanzahl	2	Wert

WRITE ($CB)

Der WRITE-Aufruf überträgt eine angegebene Anzahl von Bytes vom Pufferbereich auf eine angegebene Diskettendatei. Diese Daten werden

von der momentanen Position innerhalb der Datei an auf die Diskette geschrieben. Dann wird die Position berichtigt, indem zur letzten Position die Anzahl der tatsächlich übertragenen Bytes hinzugezählt wird. ProDOS teilt der Datei nach Bedarf zusätzliche Blöcke zu, und die Markierung für das Dateiende wird hinausgeschoben, wenn sie angetroffen wird.

Der WRITE-Aufruf hat die folgenden vier Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 4 sein)
1.	Referenznummer	1	Wert
2.	Datenpuffer	2	Zeiger
3.	Anforderungsanzahl	2	Wert
4.	Übertragungsanzahl	2	Resultat

Wenn kein Fehler aufgetreten ist, stimmt die Übertragungsanzahl mit der Anforderungsanzahl überein.

CLOSE ($CC)

Der CLOSE-Aufruf muß bei allen offenen Dateien verwendet werden. Dieser Aufruf überträgt alle Daten des Dateipuffers, die noch nicht in die Datei geschrieben worden sind, auf die Diskette, löst den Datei-Kontrollblock und den I/O-Puffer auf und bringt den Verzeichniseintrag der Datei auf den letzten Stand. Er gibt auch die der Datei zugewiesene Referenznummer wieder frei.

Wenn der in einem solchen Aufruf angegebene Parameter für die Referenznummer auf 0 gesetzt ist, sind alle offenen Dateien von dieser Stufe an geschlossen worden. Sie können die Stufe, auf der Dateien geöffnet werden, selbst festlegen, indem Sie den Wert der Speicherstelle für die Stufe einer Datei auf der System-Global-Seite ($BFD8) ändern, bevor Sie Ihre Datei öffnen. Diese Eigenschaft kann dazu verwendet werden, Dateien gruppenweise mit einem einzigen Befehl zu schließen. Ihr Programm könnte zum Beispiel zwei Dateien auf Stufe 0, zwei auf Stufe 1 und drei auf Stufe 2 öffnen. Wenn Sie dann den Wert auf der System-Global-Seite auf 1 setzen und einen CLOSE-Aufruf mit der Referenznummer 0 ausführen, werden alle Dateien der Stufen 1 und 2 geschlossen, während die Dateien der Stufe 0 offen bleiben.

Der CLOSE-Aufruf führt genau die gleichen Funktionen wie die CLOSE-Anweisung in BASIC durch.

Der CLOSE-Aufruf hat nur einen Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 1 sein)
1.	Referenznummer	1	Wert

FLUSH ($CD)

Bei einem FLUSH-Aufruf werden alle neuen Daten im I/O-Puffer einer Datei in die Datei geschrieben, und der Dateieintrag im Verzeichnis wird berichtigt. Dieser Befehl hat nur den Parameter für die Referenznummer der Datei. Wenn die Referenznummer gleich 0 ist, sind die Puffer aller Dateien von dieser Stufe an auf die Diskette geschrieben worden.

Der FLUSH-Aufruf führt genau die gleichen Funktionen wie der FLUSH-Aufruf in BASIC durch.

Der FLUSH-Aufruf hat nur einen Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 1 sein)
1.	Referenznummer	1	Wert

SET_MARK ($CE)

Der SET_MARK-Aufruf verändert die gegenwärtige Position (Marke) innerhalb einer Datei. Die neue Position ist ein absoluter Wert relativ zum Anfang der Datei. Diese neue Position darf das Dateiende nicht überschreiten.

Der SET_MARK-Aufruf hat die folgenden beiden Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 2 sein)
1.	Referenznummer	1	Wert
2.	Position	3	Wert

GET_MARK ($CF)

Der GET_MARK-Aufruf gibt den Wert der momentanen Position (Marke) innerhalb einer offenen Datei an. Diese Marke besteht immer

aus der Position, an der die nächste Lese- oder Schreiboperation anfängt. (Sie kann mit dem SET_MARK-Aufruf geändert werden.)

Der GET_MARK-Aufruf hat die folgenden beiden Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 2 sein)
1.	Referenznummer	1	Wert
2.	Position	3	Ergebnis

SET_EOF ($D0)

Der SET_EOF-Aufruf setzt das Ende einer Datei (EOF) auf den angegebenen Wert. Wenn das angegebene EOF kleiner als das bisherige EOF ist, werden alle Blöcke, die hinter dem neuen Wert liegen, für anderweitige Benutzung freigegeben. Die momentane Position innerhalb der Datei wird auch entsprechend zurückverlegt, wenn Sie hinter der neuen EOF-Marke liegt. Liegt die neue EOF-Marke hinter der bisherigen, werden der Datei keine neuen Blöcke zugewiesen. Das geschieht nur, wenn Daten dorthin geschrieben werden.

Der SET_EOF-Aufruf hat die folgenden beiden Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 2 sein)
1.	Referenznummer	1	Wert
2.	Dateiende (EOF)	3	Wert

GET_EOF ($D1)

Der GET_EOF-Aufruf gibt die maximale Anzahl Bytes an, die von einer Datei gelesen werden können. Dieser Wert wird im Verzeichniseintrag der Datei abgelesen. Er bleibt so lange gültig, bis das Dateiende mit dem SET_EOF-Aufruf geändert oder bei einem WRITE-Aufruf durch Einfügen neuer Daten in die Datei hinausgeschoben worden ist.

Der GET_EOF-Aufruf hat die folgenden beiden Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 2 sein)
1.	Referenznummer	1	Wert
2.	Dateiende (EOF)	3	Ergebnis

SET_BUF ($D2)

Mit dem SET_BUF-Aufruf können Sie die Adresse des I/O-Puffers einer offenen Datei ändern. Das MLI überprüft die System- Bitabbildung, um nachzuschauen, ob der neue Pufferbereich noch nicht belegt ist, bevor es den Inhalt des alten Puffers in den neuen kopiert. Der neue Puffer ist 1024 Bytes lang und beginnt an einem Seitenanfang im Speicher.

Der SET_BUF-Aufruf hat die folgenden beiden Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 2 sein)
1.	Referenznummer	1	Wert
2.	I/O-Puffer	2	Zeiger (muß ein Vielfaches von $100 sein)

GET_BUF ($D3)

Der GET_BUF-Aufruf gibt die Adresse des I/O-Puffers an, der zur Zeit für die in der Referenznummer angegebene Datei benutzt wird.

Der GET_BUF-Aufruf hat die folgenden beiden Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 2 sein)
1.	Referenznummer	1	Wert
2.	I/O-Puffer	2	Ergebnis

Systemaufrufe

Jeder MLI-Aufruf, der kein Datei- oder Hausverwaltungsaufruf ist, wird als Systemaufruf bezeichnet. Das ProDOS-MLI enthält folgende fünf Systemaufrufe:

ALLOC_INTERRUPT
DEALLOC_INTERRUPT
READ_BLOCK
WRITE_BLOCK
GET_TIME

Diese Routinen werden dazu verwendet, Unterbrechungsprogramme zu installieren und wieder zu entfernen, Datum und Uhrzeit zu lesen und angegebene Blöcke auf einer Diskette zu lesen oder auf sie zu schreiben.

ALLOC_INTERRUPT ($40)

Der ALLOC_INTERRUPT-Aufruf fügt die Adresse einer Unterbrechungsroutine in die Tabelle mit dem Unterbrechungsvektor ein. Sie sollten sich vergewissern, daß Sie diesen Aufruf durchgeführt haben, bevor Sie ein angeschlossenes unterbrechungsgetriebenes Gerät einschalten. Sie sind dafür verantwortlich, daß sich die Routine an der Stelle befindet, die Sie angegeben haben.

Die vom ALLOC_INTERRUPT-Aufruf zurückgegebene Unterbrechungszahl ist eine Zahl von 1 bis 4. Diese Zahl steht für die Priorität dieser Unterbrechungsroutine. Prioritäten werden in der Reihenfolge vergeben, in der die Routinen in die Tabelle mit dem Unterbrechungsvektor eingefügt worden sind. Wenn eine Unterbrechung eintritt, ruft das ProDOS-Unterbrechungsprogramm nacheinander jede einzelne in der Tabelle mit dem Unterbrechungsvektor installierte Routine auf, und zwar in der Reihenfolge von 1 bis 4. Es stoppt bei der ersten, die diese spezielle Unterbrechung beansprucht. Falls keine Routine in der Tabelle die Unterbrechung beansprucht, wird Ihr Programm abstürzen, und Sie müssen neu laden.

Der ALLOC_INTERRUPT-Aufruf hat die folgenden beiden Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 2 sein)
1.	Unterbrechungszahl	1	Ergebnis
2.	Adresse des Unterbrechungsprogramms	2	Zeiger

DEALLOC_INTERRUPT ($41)

Der DEALLOC_INTERRUPT-Aufruf entfernt einen Eintrag aus der Tabelle mit dem Unterbrechungsvektor. Sie müssen bei diesem Aufruf die im Ergebnisparameter des ALLOC_INTERRUPT-Aufrufs mitgeteilte Unterbrechungszahl als Parameter angeben. Speichern Sie diese Zahl für den zukünftigen Gebrauch, wenn Ihr Programm Unterbrechungsroutinen installiert und wieder entfernt. Denken Sie daran, daß Sie das unterbrechungsgetriebene Gerät vor der Durchführung dieses Aufrufs ausschalten müssen. Tun Sie das nicht, dann entsteht ein Systemfehler, wenn das Gerät eine Unterbrechung erzeugt, nachdem es aus der Tabelle gestrichen wurde.

Der DEALLOC_INTERRUPT-Aufruf hat nur einen Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 1 sein)
1.	Unterbrechungszahl	1	Wert

READ_BLOCK ($80)

Der READ_BLOCK-Aufruf liest einen logischen Datenblock vom angegebenen Diskettengerät. Bei diesem Befehl werden immer 512 Bytes von der Diskette gelesen, also muß Ihr Puffer mindestens so lang sein.

Der READ_BLOCK-Aufruf hat folgende drei Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 3 sein)
1.	Einheitsnummer	1	Wert
2.	Datenpuffer	2	Zeiger
3.	Blocknummer	2	Zeiger

WRITE_BLOCK ($81)

Der WRITE_BLOCK-Aufruf schreibt 512 Bytes (einen Block) aus dem Speicher Ihres Apple auf eine Diskette. Da mit diesem Befehl immer 512 Bytes Daten übertragen werden, muß Ihr Pufferbereich mindestens so lang sein. Sonst übertragen Sie das, was sich auch immer in diesem Speicherbereich befinden mag, auf die Diskette.

Der WRITE_BLOCK-Aufruf hat die folgenden drei Parameter:

	Inhalt	Länge	Typ
0.	Parameteranzahl	1	Wert (muß gleich 3 sein)
1.	Einheitsnummer	1	Wert
2.	Datenpuffer	2	Zeiger
3.	Blocknummer	2	Zeiger

Mit den Aufrufen READ_BLOCK und WRITE_BLOCK können Sie direkt auf den Inhalt eines jeden logischen Blocks auf einer Diskette zugreifen. Diese Befehle werden normalerweise von Dienstprogrammen benutzt, und Sie sollten vorsichtig mit ihnen umgehen. Die Dateistruktur von ProDOS kann beschädigt werden, wenn bei einen WRITE_BLOCK-

Befehl notwendige Informationen durch andere Daten überschrieben werden.

Sie können diese Aufrufe auch dazu verwenden, Blöcke auf Disketten zu lesen oder zu schreiben, die unter DOS 3.3 formatiert sind. Unter DOS 3.3 sind Daten mit Spur- und Sektornummer abgespeichert. Um zu bestimmen, welche ProDOS-Blocknummer einer vogegebenen Spur- und Sektornummer unter DOS 3.3 entspricht, können Sie folgende Formel benutzen:

Block = (8 * Spur) + Sektorausgleich

Der Wert des Sektorausgleichs wird durch folgende Tabelle bestimmt:

Sektor:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Sektorausgleich:	0	7	6	6	5	5	4	4	3	3	2	2	1	1	0	7
Blockhälfte:	1	1	2	1	2	1	2	1	2	1	2	1	2	1	2	2

Die Zeile für die Blockhälfte teilt Ihnen mit, ob sich der Sektor in der ersten oder der zweiten Hälfte des Blocks befindet.

Wenn Sie zum Beispiel den READ_BLOCK-Aufruf dazu benutzen möchten, einen Datenblock auf einer DOS 3.3-Diskette, angefangen bei Spur 5, Sektor 6, zu lesen, erhalten Sie die korrekte Blocknummer über folgende Berechnung. Zuerst multiplizieren Sie die Spurnummer mit 8 und erhalten die Zahl 40. Zu dieser Zahl addieren Sie den Sektorausgleich, der nach der Karte gleich 4 ist, und erhalten als Ergebnis die Zahl 44. Die Zeile für die Blockhälfte sagt uns, daß Sektor 6 in den letzten 256 Bytes des Blocks liegt (DOS 3.3-Sektoren enthalten nur 256 Bytes).

GET_TIME ($82)

Der GET_TIME-Aufruf gibt das Systemdatum und die Systemuhrzeit an. Dieser Aufruf hat keine Parameter. Er ruft eine Uhr-Kalender-Routine auf (wenn sie installiert ist), die das gegenwärtige Datum und die Uhrzeit in die vom System dafür vorgesehenen Speicherstellen ablegt. ProDOS schaut beim Laden von selbst nach, ob sich eine „Thunder-Clock"-Karte der Firma Thunderware in einem Steckplatz befindet. Wenn es eine solche Karte findet, ist eine Uhr-Kalender-Routine automatisch installiert.

Das Systemdatum ist in den Stellen $BF91 und $BF90 in der Reihenfolge Jahr, Monat, Tag gespeichert. Die Systemuhrzeit ist in den Stellen $BF93

und $BF92 gespeichert. In der Speicherstelle $BF93 befinden sich die Stunden und bei $BF92 die Minuten. Das Format der Daten ist das gleiche wie bei den Parametern für Datum und Uhrzeit, Herstellungszeitpunkt und Modifikationszeitpunkt.

Kapitel 11 ProDOS und der Apple II

Dieses Kapitel besteht aus zwei Teilen. Der erste Teil enthält eine kurze Beschreibung, wie ein ProDOS-Systemprogramm aussieht und auf welche Hilfsmittel es zurückgreifen kann. Im zweiten Teil wird der Monitor besprochen, ein Programm, das als Editor für den Speicher des Apple betrachtet werden kann. Mit dem Monitor können Sie direkt den Inhalt eines beliebigen Speicherbereichs in Ihrem Apple beeinflussen und Programme in Maschinensprache schreiben; dabei haben Sie noch alle im Direktmodus gültigen ProDOS-Befehle zur Verfügung.

ProDOS-SYSTEMPROGRAMME

Unter ProDOS ist ein Programm ein Systemprogramm, wenn es Aufrufe an das MLI enthält und folgenden drei Regeln genügt:

1. Das Programm muß Anweisungen enthalten, um sich selbst von seiner Ladeposition zu seiner Ausführungsposition verlagern zu können, falls das notwendig sein sollte.
2. Es muß die System-Global-Seite auf den neuesten Stand bringen und Versionsnummern an den dafür vorgesehenen Stellen auf der System-Global-Seite ablegen können.
3. Es muß die Programmausführung an ein anderes Systemprogramm übergeben können.

Diese Bedingungen werden im nächsten Kapitel genauer erklärt. Abgesehen von diesen Voraussetzungen bleiben alle Aspekte eines ProDOS-Systemprogramms Ihnen überlassen. Es kann sich bei dem Programm um ein Spielprogramm, ein Berechnungsprogramm oder um irgend ein anderes Programm handeln.

Anforderungen an ein Systemprogramm

Jedes Systemprogramm wird im Speicher an die Stelle $2000 geladen. Wenn das System gestartet wird, wird die erste Datei auf der Ladediskette

mit dem Namen xxx.SYSTEM (wobei xxx eine beliebige gültige Zeichenkette sein kann) und dem Typ $FF (SYS) in den Speicher geladen. Dieses Programm kann dann eine beliebige Datei vom Typ $FF laden, um sich von ihr ablösen zu lassen.

Nachdem ein Systemprogramm an die Stelle $2000 geladen worden ist, kann es zur Ausführung an irgendeine Stelle zwischen $0800 und $BEFF verlegt werden. Das muß mit Anweisungen geschehen, die im Programm selbst enthalten sind.

Die zweite Anforderung an ein Systemprogramm ist, daß es die System-Global-Seite folgendermaßen auf einen neuen Stand bringen muß:

1. In der System-Bitabbildung ($BF58 bis $BF6F) muß der Speicherbereich, in dem sich das Programm befindet, als belegt markiert werden.
2. Die Nummer der frühesten Version des MLI, mit der Ihr Programm lauffähig ist, muß an der Stelle $BFFC abgelegt werden.
3. Die Versionsnummer Ihres Systemprogramms muß an der Stelle $BFFD abgelegt werden.

Schließlich müssen alle Systemprogramme in der Lage sein, auf ein anderes Systemprogramm umzuschalten. Es ist wichtig, daß sich der Code, der diese Operation durchführt, an einer Stelle befindet, wo er beim Laden des neuen Programms nicht überschrieben werden kann. Das bedeutet, daß er sich im allgemeinen unterhalb von $2000 befindet. Zum Umschalten zwischen ProDOS-Systemprogrammen wird folgende Methode empfohlen:

1. Schließen Sie alle offenen Dateien. Das ist eine kluge Vorsichtsmaßnahme, die sehr zu empfehlen ist.
2. Fragen Sie den Anwender nach dem Namen des gewünschten Systemprogramms. (Dieser Schritt kann möglicherweise ausgelassen werden, wenn der Anwender nichts mit der Entscheidung zu tun hat, auf welches Programm umgeschaltet werden soll.)
3. Öffnen Sie die Datei, die das betreffende Systemprogramm enthält. Bestimmen Sie die Länge der Datei mit dem GET_EOF-Aufruf des MLI, so daß Sie wissen, wieviel Bytes einzulesen sind.
4. Geben Sie den Speicherbereich frei, der von Ihrem Systemprogramm belegt werden soll, indem Sie die System-Bitabbildung neu setzen.
5. Lesen Sie die gewünschte Datei ab Speicherstelle $2000 ein.

6. Schließen Sie die Datei.
7. Legen Sie den Pfadnamen des gewünschten Programms bei $0280 ab. (Erinnern Sie sich, daß vor dem Pfadnamen ein Byte mit der Länge des Pfadnamens stehen muß.) Das ist die Stelle, an der ProDOS den Pfadnamen der zuletzt geladenen Datei speichert.
8. Führen Sie eine JMP-Anweisung zur Adresse $2000 durch, um mit der Ausführung des neuen Programms zu beginnen.

Die Schritte 3 bis 8 bleiben immer gleich. Wenn im MLI beim Einlesen des neuen Programms ein Fehler auftritt, sollte Ihr Programm den Bildschirm löschen und den Anwender auffordern, das System neu zu starten, falls der Fehler nicht zu beheben ist.

Hilfsmittel für Systemprogramme

Bei einem Apple II liegt der Stapelspeicher (Stack) auf Seite $01 des Speichers, und zwar vom höchstwertigen Byte der Seite an abwärts. Systemprogrammen steht diese Seite voll zur Verfügung. Nur die untersten 16 Bytes werden auch vom Unterbrechungsprogramm des MLI benutzt. Das Unterbrechungsprogramm rettet diese 16 Bytes und speichert sie neu, sobald mehr als 75 % des Stapelspeichers voll sind. Wenn Sie es mit einem Systemprogramm zu tun haben, das mit Unterbrechungen arbeitet, und wenn Geschwindigkeit und Effizienz bei der Behandlung von Unterbrechungen Vorrang haben sollen, sollten Sie den von Ihrem Systemprogramm benutzten Bereich des Stapelspeichers auf die oberen drei Viertel einschränken. Dann braucht das Unterbrechungsprogramm nicht den unteren Teil des Stapelspeichers zu retten und neu abzuspeichern.

Wenn ein Apple IIe eine zusätzliche Bank von 64K RAM Speicherplatz hat, wird dieser zusätzliche Speicherbereich als ProDOS-Datenträger namens RAM konfiguriert und als Steckplatz 0, Laufwerk 2 behandelt. Auf diesen zusätzlichen Speicherbereich kann nicht mit einem MLI-Aufruf oder einer BASIC-Anweisung direkt zugegriffen werden, sondern ProDOS greift auf ihn wie auf eine Diskette zu. Der zusätzliche Speicherbereich in einem Apple IIc wird genauso behandelt.

Die System-Global-Seite (Seite $BF des Speichers) enthält die globalen Variablen, in denen Änderungen während des Ablaufs des Systems festgehalten werden. Einige dieser globalen Variablen dürfen von Systemprogrammen benutzt werden, einige sind nur für Informationszwecke da und sollten nicht geändert werden, und andere werden für interne Abläufe verwendet und sollten erst recht nicht verändert werden. Der Inhalt der System-Global-Seite ist in Anhang K genauer beschrieben.

PRODOS UND DER MONITOR

Der Monitor ist ein Kontrollprogramm in Maschinensprache, das sich permanent im „Nur-Lese"-Speicher (ROM) Ihres Apple befindet. Mit ihm können Sie den Inhalt von Speicherstellen und von Registern überprüfen oder verändern oder sich ganze Bereiche des Speichers ausgeben lassen. Sie können Datenblöcke vergleichen oder von einer Adresse zu einer anderen verschieben. Um es allgemein zu sagen: Der Monitor wirkt als Schlüssel zur internen Arbeitsweise Ihres Apple-Computers.

Der Monitor ist ein Werkzeug, mit dem Sie Ihren Apple erkunden und kontrollieren können. Alle im Direktmodus des BASIC gültigen ProDOS-Befehle arbeiten auch vom Monitorprogramm aus (die anderen ProDOS-Befehle sind hier nicht gültig). Mit dem Monitor haben Sie einen direkten Zugang zum MLI. Sie können damit Maschinensprache-Programme, die das MLI aufrufen, direkt in den Speicher eingeben. Sie können auch die Daten für die Parameter eingeben, wenn sie wollen. Weil Sie die ProDOS-Befehle hier zur Verfügung haben, können Sie das Ergebnis Ihrer Arbeit mit dem BSAVE-Befehl abspeichern.

In den Monitor gehen und den Monitor verlassen

Wenn Sie vom BASIC-Modus in den Monitormodus überwechseln wollen, können Sie das mit dem Befehl CALL −151 tun. Die eckige Klammer, die das BASIC-Prompt darstellt, verschwindet und wird durch das Stern-Prompt des Monitors ersetzt. Wenn Sie ins BASIC zurückgehen wollen, drücken Sie einfach die Control-Reset-Tasten oder Control-C und die Return-Taste.

Mit dem Monitor den Speicher überprüfen

Mit dem Monitor können Sie jeden Speicherbereich überprüfen, der aus einem einzelnen Byte, einem ,,Wort" aus acht Bytes oder aus einem Block von beliebiger Größe besteht. Ein einzelnes Byte können Sie überprüfen, indem Sie im Monitormodus seine Speicheradresse in hexadezimaler Form eingeben und die Return-Taste drücken. Gehen Sie mit CALL −151 in den Monitormodus, und geben Sie zum Beispiel die Adresse der Speicherstelle FF69 ein. Der Monitor wird mit

```
FF69 - A9
```

antworten. Das ist die Speicheradresse, die Sie angegeben haben, und deren Inhalt in hexadezimaler Form. Der Monitor unterhält einen Zeiger auf die Position, an der Sie sich gerade befinden. Er ändert ihn nach jeder neu eingegebenen Adresse.

Ein Wort aus acht Bytes des Speichers können Sie sich ausgeben lassen, indem Sie einfach die Return-Taste drücken. Wenn Sie zuletzt auf ein einzelnes Byte zugegriffen haben, werden Sie nach dem Drücken der Return-Taste weniger als acht Bytes sehen. Der Monitor gibt nämlich nur den Rest des Wortes aus, bei dem Sie sich befinden. Wenn Sie die Return-Taste ein zweites Mal drücken, gibt der Monitor die acht Bytes aus, die auf das letzte Byte folgen, das er Ihnen vorher gezeigt hat. Sie können sich den Unterschied zwischen den beiden Ergebnissen leicht erklären, weil der Monitor bei der Ausgabe eines vollständigen Wortes immer die Startadresse angibt. Fehlt die Startadresse, zeigt er Ihnen nur den Teil eines Wortes.

Wenn Sie zum Beispiel FF69 eingeben und dreimal die Return-Taste drücken, sehen Sie folgende Ausgabe:

```
*FF69
FF69 - A9
*
AA 85 33 20 67 FD
*
FF70 - 20 CD FF 20 A7 FF 84 34
```

Das kann eine effektive Methode zum Anschauen aufeinanderfolgender Blöcke sein, wenn Sie den Speicher nach einer bestimmten Zahlenfolge absuchen.

Einen Block des Speichers können Sie sich ausgeben lassen, indem Sie einen Adreßbereich angeben. Das geht, indem Sie Anfangs- und End-

```
F500- FC 85 27 60 18 A5 27 69
F508- 04 2C B9 F5 D0 F3 06 26
F510- 90 18 69 E0 18 2C 08 F5
F518- F0 12 A5 26 69 50 49 F0
F520- F0 02 49 F0 85 26 A5 E6
F528- 90 02 69 E0 66 26 90 D1
F530- 48 A9 00 85 E0 85 E1 85
F538- E2 68 48 38 E5 E0 48 8A
F540- E8 E1 85 D3 B0 0A 68 49
F548- FF 69 01 48 A9 00 E5 D3
F550- 85
```

Abb. 11.1: Die Ausgabe eines Speicherblocks durch den Monitor

adresse, durch einen Punkt getrennt, eingeben. Tippen Sie zum Beispiel F500.F550 im Monitormodus ein. Als Antwort sollte die Zahlenfolge in Abb. 11.1 erscheinen.

Die erste Adresse, die Sie angeben, muß kleiner als die zweite Adresse sein, oder es wird nur eine Speicherstelle ausgegeben. Wenn Sie einen Adreßbereich angeben, der zu groß ist, um auf einer Bildschirmseite ausgegeben zu werden, werden die Zeilen nach oben rollen, um Platz für neue Daten zu schaffen. Durch Drücken von Control-S können Sie das Rollen stoppen oder weitergehen lassen, damit Sie genügend Zeit haben, sich den Bildschirminhalt anzuschauen. Mit Control-S können Sie nur die Bildschirmausgabe stoppen; die Datenübertragung auf ein anderes Gerät, wie zum Beispiel auf einen Drucker, geht weiter.

Mit einer weiteren Form dieses Befehls können Sie vorwärts bis zu einer angegebenen Adresse lesen. Weil der Monitor sich merkt, bei welcher Speicheradresse Sie sich befinden, können Sie nur die Endadresse angeben, indem Sie ihr einen Punkt voranstellen und die Return-Taste drükken. Wenn Sie gerade das Beispiel von oben ausprobiert haben, geben Sie .F600 ein, und drücken Sie die Return-Taste. Der Inhalt der Speicherstellen F551 bis F600 erscheint auf dem Bildschirm. Wenn Sie eine Endadresse angeben, die kleiner als die Adresse ist, bei der Sie sich gerade befinden, gibt der Monitor nur das nächste Byte aus.

Mit dem Monitor können Sie auch den Inhalt der Register überprüfen. Das geht, indem Sie Control-E, gefolgt von der Return-Taste, drücken. Dann sehen Sie ein ähnliches Ergebnis wie

```
A=96 X=17 Y=01 P=00 S=98
```

Die aktuellen Werte in diesen Registern ändern sich laufend, aber die Register bleiben immer die gleichen. Von links nach rechts sind das der Akkumulator, die Indexregister X und Y, das Register für den Status des Prozessors und der Stapelzeiger. Der Monitor verändert den Inhalt dieser Register bei seinen Operationen nicht, also bleiben diese Werte konstant, solange Sie sich im Monitormodus befinden und sie nicht direkt ändern. Diese Eigenschaft kann extrem nützlich beim Absuchen eines Maschinensprache- oder Assembler-Programms auf Fehler sein, weil Sie sich den aktuellen Inhalt der Register, mit denen Sie arbeiten, anschauen können.

Mit dem Monitor den Inhalt des Speichers ändern

Sie können mit dem Monitor Speicherinhalte auch verändern. Der Monitorbefehl zum Ändern des Inhalts einer einzelnen Speicherstelle ist der

Doppelpunkt. Probieren Sie folgende Schrittsequenz aus, um dem Inhalt der Speicherstelle 0310 zu ändern:

```
*0310
0310 - 00
*:05
*0310
0310 - 05
```

Das gleiche Ergebnis erhalten Sie, wenn Sie 0310:05 eingeben. Dieser Befehl bewirkt, daß die Zahl hinter dem Doppelpunkt, an der vor dem Doppelpunkt angegebenen Stelle, abgespeichert wird.

Sie können auch mehrere Stellen auf einmal ändern. Das geht, indem Sie eine Folge von Zahlen hinter dem Doppelpunkt eingeben, wobei je zwei Zahlen durch ein Leerzeichen getrennt sind. Die Anweisung

```
0310:01 02 03 04 05 06 07 08
```

liest zum Beispiel in die Stellen 0310 bis 0317 die Zahlen 1 bis 8 ein. Die Speicherstellen werden immer von der Startadresse an in fortlaufender Reihenfolge geändert. Sie können überprüfen, ob die Speicherstellen auch wirklich geändert worden sind, indem Sie den Befehl 0310.0317 eingeben.

Den Inhalt der Register können Sie mit einer ähnlichen Prozedur austauschen. Bevor Sie jedoch den Inhalt der Register ändern können, müssen Sie sie durch Eintippen von Control-E und Return überprüfen. Danach können Sie die Werte in den Registern ändern, indem Sie einen Doppelpunkt und danach die neuen Werten eingeben.

Die Registerwerte müssen in genau der gleichen Reihenfolge eingegeben werden, wie sie Ihnen auf dem Bildschirm vorgestellt werden. Der Akkumulator kommt immer zuerst, dann das X-Register, das Y-Register, das Register für den Status des Prozessors und schließlich der Stapelzeiger. Wenn Sie nur den Inhalt des Y- Registers ändern und den Inhalt der anderen Register beibehalten wollen, müssen Sie die bestehenden Werte des Akkumulators und des X-Registers vor dem neuen Wert für das Y-Register eingeben. Für die letzten beiden Register brauchen Sie nichts einzugeben, wenn Sie sie nicht ändern wollen. Das liegt daran, daß Änderungen der Reihe nach vorgenommen werden, angefangen beim Akkumulator. Benutzen Sie also zum Beispiel die folgenden Schritte, um den Inhalt des Y-Registers zu ändern:

```
* (Control-E und die Return-Taste drücken)
A=96 X=17 Y=01 P=00 S=98
```

```
*:96 17 02
* (Control-E und die Return-Taste drücken)
A=96 X=17 Y=02 P=00 S=98
```

Wenn Sie den Inhalt des Stapelzeigers ändern wollen, müssen Sie vorher alle vier anderen Register eingeben. Wollen Sie den Inhalt des Akkumulators ändern, brauchen Sie für die anderen Register nichts einzugeben.

Mit dem Monitor Speicherbereiche kopieren und vergleichen

Mit dem Monitor können Sie auch den Inhalt eines Speicherblocks in einen anderen Block kopieren. Zu diesem Zweck müssen Sie dem Monitor die Anfangsadresse des Blocks, in dem die Kopie stehen soll, und Anfangs- und Endadresse des zu kopierenden Blocks mitteilen. Die Anfangsadresse des Blocks, in dem die Kopie stehen soll, kommt immer zuerst, dahinter das Zeichen <. Es folgen die Anfangsadresse des Blokkes, den Sie kopieren wollen, ein Punkt und die Endadresse des zu kopierenden Blocks. Schließlich fehlt noch der Buchstabe M, der anzeigt, daß es sich bei dem Befehl um eine Verschiebung (move) handelt. Wenn Sie also den Inhalt des Speicherbereichs von Adresse 0310 bis Adresse 0320 in den Bereich verschieben wollen, der bei 0350 anfängt, erreichen Sie das mit dem Befehl

```
0350<0310.0320M
```

Sie können diesen Befehl auch zum Kopieren von Teilen eines Programms verwenden, um Teile herzustellen, die ähnlich sind und nur geringfügige Änderungen erfordern.

Dieser Befehl verändert nicht den Inhalt des Speicherbereichs, von dem aus Sie verschoben haben. Er überschreibt aber den Inhalt des Bereichs, in den Sie den Block verschoben haben. Wie bei allen Bereichsbefehlen im Monitormodus wird der Befehl, falls Sie einen Bereich angeben, dessen Endadresse kleiner als die Anfangsadresse ist, als Befehl für ein einzelnes Byte behandelt. Dann wird nur ein Byte kopiert.

Der Monitor kann auch zwei Speicherblöcke vergleichen und überprüfen, ob sie den gleichen Inhalt haben. Das Format dieses Befehls ist dem Format des Befehls zum Verschieben gleich, außer daß der Buchstabe M am Ende der Befehlszeile durch den Buchstaben V für Verifizieren ersetzt wird.

Angenommen, Sie möchten überprüfen, ob Ihre Verschiebeoperation von eben erfolgreich war. Sie können sich beide Bereiche auf dem Bild-

schirm ausgeben lassen und sie mit dem Auge vergleichen, oder Sie können den Befehl zum Verifizieren benutzen. Die erste Methode ist umständlicher und anfälliger für Fehler. Um zu sehen, wie der Befehl arbeitet, geben Sie nach der Verschiebeoperation von oben den Befehl

```
0310<0350.0360V
```

ein. Wenn alles richtig verlaufen ist, wird kein Ergebnis ausgegeben, und es erscheint das nächste Stern-Prompt. Sie können sich selbst davon überzeugen, indem Sie den Inhalt der beiden Bereiche auflisten lassen und vergleichen.

Verändern Sie nun den Speicherinhalt mit einem Befehl wie 0312:12, und führen Sie die Anweisung zum Verifizieren noch einmal aus. Sie erhalten als Ergebnis eine Anzeige wie

```
0351 - 02 (12)
```

die Ihnen mitteilt, daß der Inhalt der Stelle 0351 gleich 02 ist, während der Inhalt der äquivalenten Stelle (0312) gleich 12 ist. Der Inhalt der Zieladresse wird immer zuletzt und in Klammern ausgegeben. Die angezeigte Adresse ist immer die Quelladresse.

Sie können diesen Befehl dazu verwenden, den Inhalt binärer Dateien zu vergleichen, die Sie auf einer Diskette gespeichert haben. Laden Sie die Dateien in zwei verschiedene Speicherbereiche und vergleichen Sie den Inhalt mit diesem Monitorbefehl. Das kann sehr nützlich sein, wenn Sie die Unterschiede zwischen zwei binären Dateien herausfinden wollen; denn mit dem ProDOS-Befehl zum Vergleichen von Dateien können Sie zwar herausfinden, ob sich zwei Dateien unterscheiden, nicht aber, wo sie sich unterscheiden.

Weitere Eigenschaften des Monitors

Wenn Sie ein Programm in Maschinensprache ausführen wollen, können Sie mit dem Monitor die Kontrolle an die Startadresse dieses Programms übergeben. Geben Sie die Anfangsadresse Ihres Programms und den Buchstaben G (für Go) ein. Mit dem Befehl 03D0G zum Beispiel übergeben Sie die Kontrolle an die Anweisung in der Speicherstelle 03D0, die Sie zum BASIC-Prompt zurückbringt. Wenn Sie keine Adresse angeben, versucht der Monitor die Anweisung in der Speicherstelle auszuführen, bei der er sich gerade befindet.

Sie können die Ausgabe des Monitors auf Ihren Drucker umlenken, indem Sie die Steckplatznummer der Karte angeben, die den Drucker

kontrolliert, und danach Control-P und die Return-Taste drücken. Daraufhin erfolgt die gesamte Ausgabe über den Drucker, bis Sie durch Eintippen von 0 (für Steckplatz 0), Control-P und Return die Ausgabe wieder auf den Bildschirm zurücklenken. Vergewissern Sie sich, daß Sie die richtige Steckplatznummer angegeben haben, sonst müssen Sie Ihren Computer neu starten. Auf diese Weise können Sie sich also Speicherinhalt ausdrucken lassen, wenn Sie sich ein Programm oder die Daten in einem bestimmten Speicherabschnitt betrachten wollen.

Auch die Eingabe Ihres Apple können Sie von der Tastatur auf ein anderes Gerät, wie zum Beispiel ein Diskettenlaufwerk, umlenken. Tippen Sie dazu die Steckplatznummer, Control-K und Return ein. Die Methode gleicht der oben beschriebenen Methode für das Umlenken der Ausgabe. Sie können sie auf die gleiche Art verwenden wie die BASIC-Anweisung IN#.

Wenn Sie den Befehl I eintippen und die Return-Taste drücken, erscheinen die Antworten des Monitors auf vertauschtem Hintergrund. Ihre Eingaben erscheinen weiterhin auf normalem Hintergrund, aber Sie sehen am Stern-Prompt, das auf vertauschtem Hintergrund dargestellt wird, daß Sie sich im inversen Ausgabemodus befinden. Indem Sie N eintippen und die Return-Taste drücken, kommen Sie in den normalen Ausgabemodus zurück.

Im Monitormodus können Sie hexadezimale Additionen und Subtraktionen durchführen, indem Sie einfach eintippen, was Sie berechnen wollen. Dem in der folgenden Zeile erscheinenden Ergebnis wird ein Gleichheitszeichen vorangestellt. Der Monitor kann nur Acht-Bit-Operationen und einfache Operationen mit zwei Operanden durchführen. Das folgende Beispiel zeigt, wie der Monitor das macht:

```
*10-0A
=06
*0A+05
=0F
*0A-03
=07
*0A+05-03
=07
```

Beachten Sie, daß das letzte Beispiel einen Fehler enthält. Die Ursache dafür ist, daß der Monitor nur den ersten und den letzten Operanden verwendet. Außerdem kann er nur zweistellige Berechnungen korrekt

durchführen. Bei Ergebnissen, die größer als FF sind, wird die erste Stelle abgeschnitten.

Im Monitormodus haben Sie die Möglichkeit, einen Monitorbefehl selbst zu definieren. Sie erreichen diesen Befehl, indem Sie Control-Y eintippen und die Return-Taste drücken. Dadurch wird ein Sprung zur Adresse $03F8 veranlaßt. An dieser Stelle ist gerade genug Platz, um einen Sprung zu einer weiteren Adresse durchzuführen. Diese Möglichkeit können Sie dazu verwenden, an der Stelle $03F8 einen Sprungbefehl zu der Anfangsadresse eines Maschinensprache-Programms im Speicher abzulegen, auf das Sie dann im Monitormodus einfach mit Control-Y und Return zugreifen können. Das kann sehr nützlich sein, wenn Sie ein solches Prgramm bei der Fehlersuche öfter neu starten wollen.

ProDOS-Befehle und der Monitor

Alle ProDOS-Befehle, die im Direktmodus von BASIC zur Verfügung stehen, gibt es auch im Monitormodus. Sie können sich mit dem CAT-Befehl das Inhaltsverzeichnis einer Diskette anschauen, Sie können mit dem CLOSE-Befehl offene Dateien schließen, oder Sie können irgend einen anderen gültigen ProDOS-Befehl benutzen. Von hier aus können Sie auch mit den Befehlen BSAVE und BLOAD binäre Daten von und zur Diskette übertragen, so daß Sie sie im Monitormodus leicht überprüfen können. Sie können keinen ProDOS-Befehl benutzen, der nicht im Direktmodus von BASIC gültig ist. Die Befehle OPEN, READ, WRITE, und APPEND stehen Ihnen also hier nicht zur Verfügung.

Wenn Sie versuchen, einen ProDOS-Befehl auszuführen, und dabei ein Fehler auftritt, dann verlassen Sie den Monitormodus, und Sie werden dann ans BASIC-Prompt übergeben. In diesem Fall können Sie natürlich mit CALL −151 in den Monitormodus zurückkommen. An den Daten und Programmen, die Sie im Speicher abgespeichert hatten, hat sich nichts geändert.

Der Monitor bietet Ihnen die Möglichkeit, Programme in Maschinensprache direkt einzutippen. Das ist wichtig, wenn Sie keinen Assembler besitzen. Ein Assembler besteht aus einem Programm, das Assembler-Programme in Maschinensprache-Programme übersetzt. Wenn Sie mit dem Monitor programmieren, müssen Sie anstelle der Assembler-Anweisungen die entsprechenden Maschinensprache-Anweisungen eintippen. Dazu sind eine gute Kenntnis der Assembler-Sprache und ein Handbuch erforderlich, das die Assembler-Befehle für den 6502-Mikroprozessor enthält. Sie müssen dann die Befehle selbst in Maschinensprache übersetzen, bevor Sie sie eintippen.

Der Vorteil, den der Monitor jemandem bietet, der unter ProDOS programmiert, besteht darin, daß die Maschinensprache-Programme das MLI direkt aufrufen können. Sie brauchen dazu nur die Assembler-Routine aus Kapitel 10 in Maschinensprache zu übersetzen und mit BSAVE auf einer Diskette abzuspeichern. Dann brauchen Sie nur noch die Parameterliste mit dem Monitor in den Speicher einzugeben, den Aufruf fertigzumachen und mit dem Befehl G des Monitors auszuführen. Wenn Sie genügend Zeit und Ausdauer besitzen, können Sie so alle Möglichkeiten des MLI ausnutzen, ohne einen Assembler zu verwenden.

Jedes Maschinensprache-Programm, das Sie im Monitormodus schreiben, können Sie mit dem BSAVE-Befehl auf einer Diskette speichern und mit dem BLOAD-Befehl wieder laden. Ein solches Programm können Sie auch von BASIC aus mit den Befehlen CALL und USR aufrufen. Um das richtig zu machen, müssen Sie Ihre Maschinensprache-Anweisungen in einem Bereich des Speichers ablegen, in dem sie vom BASIC nicht überschrieben werden können (wie zum Beispiel im Bereich der Seite 2 für Text und niedrigauflösende Grafik), und Sie müssen bei diesen beiden Befehlen die aktuelle Startadresse Ihrer Maschinensprache-Routine angeben.

Während die Verwendung des Monitors zugegebenermaßen nicht die einfachste Möglichkeit ist, in Maschinensprache zu programmieren, so ist sie doch eine akzeptable Alternative zum Kauf eines Assemblers. Man braucht dazu eine Menge Ausdauer und Praxis, aber durch die Erfahrungen beim Programmieren im Monitormodus werden Sie mit mehr Einsicht in die wirkliche Arbeitsweise Ihres Apple und des Betriebssystems ProDOS belohnt werden.

Anhang A
ProDOS-Dateitypen

Die folgende Tabelle enthält alle ProDOS-Dateitypen. Der Code besteht aus einer Hexadezimalzahl, die in der Verzeichnisdatei gespeichert wird. Wenn Sie sich mit den ProDOS-Befehlen CAT oder CATALOG oder dem entsprechenden Dienstprogramm eine Verzeichnisdatei ausgeben lassen, sehen Sie eine aus drei Buchstaben bestehende Erklärung und nicht den Code selbst. Bei dem IIc ist eine solche Erklärung auf ein ganzes Wort ausgedehnt worden.

Dateitypen werden zur Identifizierung des Inhalts einer Datei verwendet. Einige ProDOS-Befehle können nur auf einen bestimmten Dateityp angewendet werden. Die Befehle LOAD, SAVE und RUN arbeiten zum Beispiel nur mit Dateien vom Typ BAS zusammen.

Code	**Dateityp**
$00	typlose Datei
BAD ($01)	Datei mit beschädigten Blöcken
TXT ($04)	ASCII-Textdatei
BIN ($06)	Binärdatei
DIR ($0F)	Verzeichnisdatei
CMD ($F0)	hinzugefügte ProDOS-Befehlsdatei
$F1–$F8	vom Anwender definierter ProDOS-Dateityp 0–9
$F9	reserviert für zukünftige Zwecke
INT ($FA)	Integer-BASIC-Programmdatei
IVR ($FB)	Integer-BASIC-Variablendatei
BAS ($FC)	Applesoft-Programmdatei
VAR ($FD)	Applesoft-Variablendatei
REL ($FE)	verlegbarer Code in Maschinensprache
SYS ($FF)	ProDOS-Systemdatei
$C0–$EF	reserviert für zukünftige Verwendungszwecke

Anhang B
Fehlermeldungen der ProDOS-Dienstprogramme

Die folgenden Fehlermeldungen werden von den Dienstprogrammen auf der ProDOS-Anwenderdiskette (ProDOS User's Disk) ausgegeben. Jeder Meldung folgt eine kurze Beschreibung und ein Hinweis, was bei ihrem Erscheinen zu tun ist.

CAN'T DELETE DIRECTORY FILE *(kann Verzeichnisdatei nicht löschen)*

Diese Fehlermeldung bedeutet, daß eine DOS-Datei, die Sie nach ProDOS übertragen wollen, den gleichen Namen wie eine ProDOS-Verzeichnisdatei hat. Sie können diese Datei mit dem CONVERT-Dienstprogramm so nicht übertragen. Zuerst müssen Sie entweder die ProDOS-Verzeichnisdatei oder die DOS-Datei umbenennen, bevor Sie einen neuen Versuch unternehmen.

CAN'T TRANSFER DIRECTORY FILE *(kann Verzeichnisdatei nicht übertragen)*

Das passiert, wenn Sie eine ProDOS-Verzeichnisdatei auf eine DOS-Diskette übertragen wollen. Da es unter DOS keinen Dateityp gibt, der die gleiche Funktion wie eine ProDOS-Verzeichnisdatei hat, erlaubt Ihnen das CONVERT-Dienstprogramm diese Übertragung nicht.

DIRECTORY ALREADY EXISTS *(Verzeichnis existiert schon)*

Diese Fehlermeldung erscheint, wenn Sie eine Verzeichnisdatei anzulegen versuchen, die den gleichen Namen wie eine bereits existierende Verzeichnisdatei hat, oder wenn sie versuchen, eine Datei in eine Verzeichnisdatei umzukopieren. Im ersten Fall müssen Sie einen anderen Namen nehmen, im zweiten Fall sollten Sie den Pfadnamen überprüfen und den Fehler korrigieren.

DIRECTORY EXPECTED *(Verzeichnis erwartet)*

Diese Fehlermeldung erscheint, wenn Sie an einer Stelle, an der ein Verzeichnisname eingegeben werden muß, einen Dateinamen eingegeben haben. Das bedeutet, daß die Reihenfolge in Ihrem Pfadnamen nicht stimmt oder daß Sie sich beim Eintippen des Pfadnamens verschrieben haben. Sie können sich die Verzeichnisse im Pfadnamen mit dem CAT-Befehl auflisten lassen, um sich den Typ und die richtige Schreibweise der Namen anzuschauen.

DIRECTORY NOT EMPTY *(Verzeichnis ist nicht leer)*

Um zu verhindern, daß Sie eine Verzeichnisdatei löschen, die voll von wichtigen Dateien ist, erlaubt ProDOS Ihnen nicht, eine Verzeichnisdatei zu löschen, die noch andere Dateien enthält. Wenn Sie sehen wollen, welche Dateien das Verzeichnis, das Sie zu löschen versucht haben, enthält, können Sie es sich auflisten lassen. Sie müssen alle in dem Verzeichnis enthaltenen Dateien löschen, bevor Sie die Verzeichnisdatei selbst löschen können.

DIRECTORY NOT FOUND *(Verzeichnis nicht gefunden)*

Wenn ProDOS aus irgend einem Grunde das von Ihnen angeforderte Verzeichnis nicht findet, erscheint diese Meldung. Sie sollten Ihren Pfadnamen überprüfen, um zu sehen, ob Sie ihn richtig eingetippt haben. Trifft das zu, sollten Sie durch Auflisten (mit dem CAT-Befehl) aller betroffenen Verzeichnisse noch einmal überprüfen, ob Sie die Unterverzeichnisse im Pfadnamen in der richtigen Reihenfolge angegeben haben. Wenn es immer noch nicht funktioniert, öffnen Sie die Laufwerktür, und schauen Sie nach, ob Sie die richtige Diskette eingelegt haben.

DISK II DRIVE TOO FAST *(Diskettenlaufwerk ist zu schnell)*

Diese Meldung erscheint nur beim Kopieren oder Formatieren einer Diskette. Sie bedeutet, das Diskettenlaufwerk dreht sich zu schnell, so daß ProDOS nicht in der Lage ist, die Diskette zu lesen oder auf sie zu schreiben. In diesem Fall muß die Geschwindigkeit des Laufwerks justiert werden, bevor es unter ProDOS benutzt werden kann.

DISK II DRIVE TOO SLOW *(Diskettenlaufwerk ist zu langsam)*

Diese Meldung erscheint nur beim Kopieren oder Formatieren einer Diskette. Sie bedeutet, Ihr Diskettenlaufwerk dreht sich zu langsam, so daß ProDOS nicht in der Lage ist, die Diskette zu lesen oder auf die Diskette zu schreiben. Sie müssen die Geschwindigkeit des Laufwerks justieren lassen, bevor Sie es unter ProDOS benutzen können.

DISK WRITE-PROTECTION *(Diskette ist schreibgeschützt)*

Diese Meldung erscheint, wenn ProDOS versucht hat, auf eine Diskette zu schreiben, die mit einem Schreibschutz versehen ist. Das bedeutet, daß die Schreibschutzkerbe der Diskette mit einer Lasche zugeklebt ist. Sie können die Lasche entfernen, sollten sich das aber genau überlegen. Die Lasche ist dort angebracht worden, um zu verhindern, daß auf die Diskette geschrieben werden kann – sind Sie sicher, daß Sie sie entfernen wollen?

DUPLICATE FILE NAME *(doppelter Dateiname)*

Wenn Sie diese Meldung sehen, haben Sie versucht, einer Datei einen Namen zuzuweisen, der in dem gleichen Verzeichnis schon existiert. Wenn das beim Umbenennen einer Datei vorkommt, versuchen Sie es mit einem anderen Namen. Kommt das beim Übertragen mit dem CONVERT-Dienstprogramm vor, so können Sie die Übertragung abbrechen oder fortsetzen und dabei die Datei auf der Zieldiskette mit diesem Namen löschen.

DUPLICATE VOLUME *(doppelter Diskettenname)*

Diese Fehlermeldung bedeutet: Sie haben versucht, den COMPARE VOLUME- oder den COPY VOLUME-Befehl auf zwei Disketten mit dem gleichen Namen anzuwenden.

ERROR CODE = XX *(Fehlercode = XX)*

Das ist die allgemeine Fehlermeldung der ProDOS-Dienstprogramme. Sie erscheint dann, wenn ein Fehler aufgetreten ist, der von den Dienstprogrammen nicht behandelt werden kann. Die beiden Ziffern hinter der Meldung (die durch XX repräsentiert werden) geben den Hexadezimalcode für eine der ProDOS-Fehlermeldungen an, die in Anhang C zu finden sind.

FILES DO NOT MATCH *(Dateien stimmen nicht überein)*

Diese Meldung erscheint, wenn Sie mit dem COMPARE-Befehl zwei Dateien vergleichen, die nicht identisch sind. Tritt sie beim Verifizieren einer Sicherungskopie auf, sollten Sie mit dem CAT-Befehl nachschauen, welche Dateien auf den beiden Diskette sich im Datum unterscheiden, und die ältere Datei durch die neuere ersetzen.

FILE EXPECTED *(Datei erwartet)*

Diese Fehlermeldung tritt auf, wenn Sie eine Datei löschen oder ihren Schreibschutz ändern wollen und der Pfadname auf ein Wurzelverzeich-

nis zeigt. Beides ist unter ProDOS bei einem Wurzelverzeichnis nicht möglich. Sie können das Wurzelverzeichnis einer Diskette nur löschen, indem Sie die Diskette neu formatieren. Die einzige Möglichkeit, eine Diskette mit einem Schreibschutz zu versehen, besteht darin, ihre Schreibschutzkerbe mit einer Lasche zuzukleben.

FILE LOCKED *(Datei ist schreibgeschützt)*

Diese Meldung erscheint beim Versuch, eine mit einem Schreibschutz versehene Datei zu löschen oder umzubenennen. Wenn Sie das wirklich tun wollen, müssen Sie vorher mit dem ALTER WRITE-PROTECTION-Befehl den Schreibschutz der Datei entfernen.

FILE NOT FOUND *(Datei nicht gefunden)*

Die Datei, die Sie angegeben haben, wurde nicht gefunden. Überprüfen Sie, ob Sie den richtigen Pfadnamen eingetippt haben und ob Ihnen kein Schreibfehler unterlaufen ist. Dabei kann Ihnen der CAT-Befehl behilflich sein.

FILE TOO LARGE *(Datei ist zu groß)*

Diese Meldung erscheint, wenn Sie versuchen, eine Datei zu kopieren und auf der Zieldiskette nicht genügend Platz frei ist. Sie müssen dann die Datei auf eine andere Diskette kopieren oder mehr Platz auf der Diskette schaffen, indem Sie einige Dateien löschen, die Sie nicht mehr brauchen.

I/O ERROR *(Eingabe- oder Ausgabefehler)*

Das ist eine allgemeine Fehlermeldung, die in vielen verschiedenen Situationen auftreten kann. Sie bedeutet, daß ProDOS aus irgend einem Grund die Diskette, die Sie angegeben haben, nicht lesen kann. Sie können die Laufwerktür offen gelassen haben, oder das Laufwerk kann leer sein. Es kann sein, daß Ihre Diskette nicht formatiert ist oder daß sich das Verbindungskabel zum Laufwerk gelockert hat. Sie sollten die einfachen Sachen zuerst überprüfen, um zu sehen, ob Sie das Problem beheben können. Geht das nicht, ist Ihre Diskette vielleicht beschädigt, und Sie sollten es mit einer anderen Diskette versuchen. Wenn sich Ihr Problem so nicht lösen läßt, sollten Sie sich an Ihren Händler wenden.

ILLEGAL CHARACTER *(unerlaubtes Zeichen)*

Diese Fehlermeldung tritt auf, wenn Sie bei der Angabe einer Datei oder eines Pfadnamens ein unerlaubtes Zeichen eingegeben haben. Schauen Sie nach, ob Sie den Fehler finden können. Gegebenenfalls sollten Sie

sich in Kapitel 4 die Bedingungen für einen Pfadnamen noch einmal durchlesen.

ILLEGAL WILDCARD *(unerlaubtes Joker-Zeichen)*

Diese Meldung erscheint, wenn Sie bei einem Pfadnamen mehr als ein Joker-Zeichen benutzt haben. Wollen Sie ein Verzeichnis auflisten, so dürfen Sie ein Joker-Zeichen nur an der ersten Position und nur als das einzige Zeichen benutzen. Um diesen Fehler zu korrigieren, müssen Sie den Pfadnamen im richtigen Format neu eingeben.

INSUFFICIENT MEMORY TO RUN PROGRAM *(zu wenig Speicherplatz, um das Programm laufen zu lassen)*

Dieser Fehler tritt nur dann auf, wenn Ihr Apple weniger als 64K RAM Speicherplatz hat. In diesem Fall können Sie ohne Speichererweiterung das ProDOS-Betriebssystem nicht benutzen.

INVALID DATE *(ungültiges Datum)*

Wenn diese Fehlermeldung erscheint, haben Sie versucht, ein unmögliches Datum einzugeben. ProDOS überprüft, ob Ihre Eingaben das richtige Format haben und gibt diese Meldung aus, wenn das Monatsfeld nicht aus den ersten drei Buchstaben eines Monats (in englischer Sprache) besteht. Diese Meldung erscheint auch, wenn die Jahreszahl kleiner als 00 oder größer als 99 ist oder wenn der Tag größer ist, als für den entsprechenden Monat erlaubt ist.

INVALID DRIVE *(ungültiges Laufwerk)*

Diese Meldung bedeutet, daß Sie eine falsche Laufwerknummer eingegeben haben. Es ist nur 1 oder 2 erlaubt, da an einen Steckplatz nur zwei Laufwerke angeschlossen werden können. Geben Sie also die richtige Nummer ein, wenn Sie weitermachen wollen.

INVALID PATHNAME *(ungültiger Pfadname)*

Diese Meldung bedeutet, daß Sie bei der Eingabe des Pfadnamens ein unerlaubtes Zeichen eingetippt haben oder daß Ihr Präfix nicht stimmt. Überprüfen Sie, ob Sie einen Fehler im Pfadnamen finden können und schlagen Sie wenn nötig in Kapitel 4 nach.

INVALID SLOT *(ungültiger Steckplatz)*

Diese Meldung erscheint, wenn Sie eine Steckplatznummer angeben, die nicht im Bereich von 1 bis 7 liegt. Das sind die einzigen Steckplätze, in

denen ProDOS eine Laufwerkkontrollkarte zuläßt. Diese Meldung wird nicht ausgegeben, wenn Sie die Nummer eines Steckplatzes aus dem Bereich angeben, in dem sich keine Laufwerkkontrollkarte befindet. (In diesem Fall sehen Sie die Meldung NO DEVICE CONNECTED.) Geben Sie also die richtige Nummer an, und vergewissern Sie sich, daß sich eine Laufwerkkontrollkarte in diesem Steckplatz befindet.

NAME TOO LONG *(Name zu lang)*

Diese Meldung sehen Sie, wenn Sie einen ProDOS-Dateinamen aus mehr als 15 Zeichen oder einen DOS-Dateinamen aus mehr als 30 Zeichen eingegeben haben (beim CONVERT-Dienstprogramm). Überprüfen Sie, ob Sie den Namen richtig eingetippt haben.

NO DATA IN FILE *(keine Daten in der Datei)*

Diese Meldung bedeutet, daß sich keine Daten in der Datei befinden, die Sie zu übertragen versuchen. Wenn Sie aus irgendeinem Grund eine leere Datei benötigen, können Sie diese mit dem CREATE-Befehl anlegen.

NO DEVICE CONNECTED *(kein Gerät angeschlossen)*

Diese Meldung bedeutet, daß an dem Steckplatz, den Sie angegeben haben, kein Laufwerk angeschlossen ist, oder daß es nicht eingeschaltet ist, wenn es sich um ein Festplattenlaufwerk handelt. Wenn Sie sich gerade etwas ausdrucken lassen, bedeutet die Meldung, daß der Drucker nicht angeschlossen ist oder die Druckerkarte sich nicht in dem Steckplatz befindet, den Sie angegeben haben. Sie können mit der DISPLAY SLOT ASSIGNMENTS-Option überprüfen, ob Sie die richtige Steckplatznummer angegeben haben. Wenn Sie das nicht weiterbringt, überprüfen Sie, ob sich eine Verbindung gelockert hat.

NO DIRECTORY *(kein Inhaltsverzeichnis)*

Diese Meldung bedeutet, daß eine eingelegte Diskette nicht formatiert oder für ein anderes Betriebssystem formatiert ist. Wenn sie unter DOS 3.3 formatiert ist, können Sie das DOS-FIB-Dienstprogramm benutzen oder die Dateien nach ProDOS konvertieren. Wenn sie unformatiert ist, können Sie sie zur Verwendung unter ProDOS herrichten, indem Sie sie mit dem FORMAT A VOLUME-Befehl formatieren.

NO PRINTER CONNECTED *(kein Drucker angeschlossen)*

Sie sehen diese Meldung, wenn Sie den Drucker als Standardausgabegerät angegeben haben und kein Drucker an Ihr System angeschlossen ist. Schließen Sie entweder den Drucker an, oder setzen Sie die Standardausgabe auf den Monitor zurück, bevor Sie weitermachen.

NO ROOM ON VOLUME *(kein Platz auf der Diskette)*

Diese Meldung bedeutet, daß auf der Zieldiskette nicht genügend Platz für die Dateien ist, die Sie von DOS nach ProDOS oder von ProDOS nach DOS konvertieren wollen. Sie müssen entweder Dateien auf der Diskette löschen oder eine andere Diskette benutzen.

NOT A DOS 3.3 VOLUME *(keine DOS 3.3-Diskette)*

Diese Meldung bedeutet, daß sich in dem Laufwerk, das Sie beim CONVERT-Dienstprogramm als Laufwerk für die DOS 3.3-Diskette angegeben haben, keine DOS 3.3-Diskette befindet. Überzeugen Sie sich, ob Sie die richtige Diskette ins richtige Laufwerk gelegt haben und ob die Richtungszeile oben auf dem Bildschirm korrekt ist.

NOT A PRODOS DIRECTORY *(kein ProDOS-Inhaltsverzeichnis)*

Das bedeutet, daß sich in dem Laufwerk, das Sie beim „CONVERT"-Dienstprogramm für die ProDOS-Diskette angegeben haben, keine ProDOS-Diskette befindet. Schauen Sie nach, ob Sie die richtige Diskette ins richtige Laufwerk gelegt haben und ob die Richtungszeile oben auf dem Bildschirm stimmt.

NOT A PRODOS INTERPRETER *(kein ProDOS-Interpreter)*

ProDOS ist nicht in der Lage, die Programmdatei auszuführen, die Sie beim QUIT-Aufruf angegeben haben. Das kann daran liegen, daß der Dateiname oder der Diskettenname für ProDOS nicht gültig ist. Wenn Sie eine gültige ProDOS-Startdiskette im Laufwerk haben, können Sie mit Control-Reset aus dieser Situation herauskommen. (Bei einem IIe oder IIc müssen Sie zusätzlich noch die Taste „Offener Apfel" drücken). Dann lädt der Apple neu von der Diskette im Standardlaufwerk.

NOT A PRODOS VOLUME *(keine ProDOS-Diskette)*

Diese Meldung bedeutet, daß Sie einen ProDOS-Befehl bei einer Diskette benutzen wollen, die nicht für ProDOS formatiert ist. Das kann eine für CP/M, Pascal, oder DOS 3.3 formatierte Diskette sein, oder es kann sich um eine unformatierte Diskette handeln. Wenn Sie auf Ihrem Apple häufig verschiedene Betriebssysteme benutzen, sollten Sie sich ein Schema zum Markieren ausdenken, damit Sie die Disketten leicht zuordnen können. Einige ProDOS-Befehle (COPY A VOLUME, COMPARE VOLUMES, DETECT BAD BLOCKS und FORMAT A VOLUME) können auch bei nicht unter ProDOS formatierten Disketten verwendet werden.

NOT THE SAME DEVICE TYP *(nicht der gleiche Gerätetyp)*

Das bedeutet, daß Sie eine Diskette auf eine Festplatte kopieren oder mit einer Festplatte vergleichen wollen. ProDOS erlaubt Ihnen das nicht wegen der vielen Fehlermöglichkeiten. Sie müssen den COPY FILES-Befehl im Menü der Datei-Dienstprogramme benutzen. Diese Meldung kann auch auftauchen, wenn Sie eine Voreinstellung machen, um Sie vor dem gleichen Fehler zu bewahren. Der Fehler kann nur dann auftreten, wenn Sie mit den ProDOS-Dienstprogrammen ganze Disketten oder eine Festplatte bearbeiten.

NOT THE SAME DIRECTORY *(nicht das gleiche Verzeichnis)*

Sie haben versucht, eine Datei umzubenennen, die in dem von Ihnen angegebenen Verzeichnis nicht vorkommt. Überprüfen Sie den Pfadnamen und das Präfix, bevor Sie es noch einmal versuchen.

PATH NOT FOUND *(Pfad nicht gefunden)*

Das bedeutet, daß die von Ihnen angegebene Diskette existiert, aber der Pfadname nicht korrekt ist. Überprüfen Sie, ob Sie ihn richtig geschrieben haben. Wenn das den Fehler nicht behebt, listen Sie die Dateien mit dem CAT-Befehl, und überprüfen Sie, ob alle Dateien existieren, die Sie im Pfadnamen angegeben haben.

PATHNAME TOO LONG *(Pfadname zu lang)*

Das bedeutet, daß Sie bei einer Transfer-Operation einen Pfadnamen eingetippt haben, der länger als 128 Zeichen oder länger als 64 Zeichen ist, wenn Sie das Präfix setzen wollten. Schauen Sie sich den Pfadnamen an, um den Fehler herauszufinden, und berichtigen Sie ihn vor dem nächsten Versuch.

PATHNAME INDICATE SAME FILE *(Pfadname zeigt die gleiche Datei an)*

Bei dieser Fehlermeldung haben Sie versucht, eine Datei auf sich selbst zu kopieren. ProDOS läßt das nicht zu. Sie müssen den Pfadnamen ändern, um der Zieldatei einen anderen Namen zu geben.

PREFIX NOT SET *(Präfix nicht gesetzt)*

Sie haben versucht, mit dem CONVERT-Dienstprogramm Dateien zu übertragen oder aus einem ProDOS-Verzeichnis aufzulisten, ohne ein

Präfix zu setzen. Sie müssen zurückgehen und das Präfix setzen, bevor Sie Ihre Dateien übertragen können.

SAME FIXED DISK *(gleiche Festplatte)*

Sie versuchen, eine Festplatte (z. B. ProFile) auf sich selbst zu kopieren oder mit sich selbst zu vergleichen. Das geht nicht. Überprüfen Sie Steckplatz- und Laufwerknummern, um festzustellen, ob Sie einen Fehler gemacht haben.

VOLUME DIRECTORY FULL *(Stammverzeichnis voll)*

Im Stammverzeichnis ist kein Platz mehr für die Datei oder das Verzeichnis, das Sie anlegen wollen. Sie müssen entweder Dateien löschen, um Platz zu schaffen, oder ein anderes Verzeichnis benutzen.

VOLUME FULL *(Diskette ist voll)*

Auf der Diskette ist kein Platz mehr. Sie müssen entweder Dateien auf der Diskette löschen, um Platz zu schaffen, oder eine andere Diskette benutzen.

VOLUME NOT FOUND *(Diskette nicht gefunden)*

ProDOS kann die Diskette mit dem Namen, den Sie angegeben haben, nicht finden. Das kann bedeuten, daß die Laufwerktür nicht geschlossen ist, oder daß sich eine falsche Diskette im Laufwerk befindet. Überprüfen Sie auch, ob Sie einen Tippfehler bei der Eingabe des Namens gemacht haben. Sie können den LIST VOLUMES-Befehl benutzen, um sicherzugehen, daß Sie den richtigen Namen angegeben haben, daß die Diskette für ProDOS formatiert ist und daß sich die richtige Diskette im Laufwerk befindet.

WILDCARD MUST BE IN FINAL NAME *(Joker-Zeichen darf nur im abschließenden Namen stehen)*

Diese Meldung bedeutet, daß Sie ein Joker-Zeichen an der falschen Stelle innerhalb eines Pfadnamens verwendet haben. Ein Joker- Zeichen darf nur im letzten Namen eines Pfadnamens stehen. Sie müssen den Fehler suchen und den Pfadnamen korrekt eingeben.

WILDCARD NOT ALLOWED *(Joker-Zeichen nicht zugelassen)*

Sie haben versucht, Joker-Zeichen bei einem Befehl zu benutzen, bei dem ProDOS das nicht zuläßt. Sie müssen den vollen Namen ohne Joker-Zeichen neu eintippen.

WILDCARD NOT PROCESSED *(Joker-Zeichen nicht verarbeitet)*

Beim Einsetzen der Zeichenfolgen für das Joker-Zeichen ist ein Pfadname zu lang geworden. Prüfen Sie die Pfadnamen nach. Wenn Sie viele lange Namen in Ihren Verzeichnissen verwendet haben, müssen Sie sie vielleicht mit dem RENAME-Dienstprogramm kürzen.

WILDCARD USE INCONSISTENT *(Verwendung der Joker-Zeichen ist inkonsistent)*

Sie versuchen Dateien unter Verwendung eines Joker-Zeichen zu kopieren oder umzubenennen, wobei die Pfadnamen der Quell- und der Zieldateien nicht zueinander passen. Sie müssen das gleiche Joker-Zeichen sowohl bei den Pfadnamen der Quelldateien als auch bei den Pfadnamen der Zieldateien verwenden. Berichtigen Sie die Pfadnamen, und versuchen Sie es noch einmal.

Anhang C

ProDOS-Fehlermeldungen

Dieser Anhang enthält alle Fehlercodes, die ProDOS zurückgeben kann, wenn es einen Befehl nicht ausführen kann. Die Spalte für die ProDOS-Meldung enthält den Text, der normalerweise beim Auftreten des entsprechenden Fehlers auf dem Bildschirm erscheint, und die Spalte für die häufigste Ursache ist vorgesehen, um Ihnen einen Tip zu geben, was den Fehler verursacht haben könnte.

Der Fehlercode wird immer in die Speicherstelle 222 ($DE) zurückgegeben. Diesen Code können Sie sich mit einer Anweisung wie

```
A = PEEK(222)
```

über die Variable A ausgeben lassen. Wenn Sie die Applesoft-Anweisung ONERR GOTO in Ihrem Programm benutzen, erscheint die ProDOS-Meldung nicht auf dem Bildschirm. Das Programm zweigt nämlich zu der Marke ab, die Sie in der ONERR-Anweisung angegeben haben, und Ihr Programm muß in der Speicherstelle 222 nachschauen, um den aufgetretenen Fehler zu bestimmen.

Als zusätzliche Hilfe zur Bestimmung der Ursache eines ProDOS-Fehlers in einem Applesoft-Programm wird die Nummer der Zeile, in der der Fehler aufgetreten ist, in die Speicherstellen 218 und 219 zurückgegeben. Aus diesen beiden Werten kann die Zeilennummer durch folgende Anweisung berechnet und in der Variablen FZ abgelegt werden:

```
FZ = PEEK(218) + (PEEK(219) * 256)
```

Code	ProDOS-Meldung	häufigste Ursache
2	RANGE ERROR *(Bereichsfehler)*	Die beim Befehl angegebenen Parameter sind zu groß oder zu klein. Eine Adresse kann zu groß sein, oder die Endadresse kann kleiner als die Anfangsadresse sein.

3	NO DEVICE CONNECTED *(kein Gerät angeschlossen)*	Am angegebenen Steckplatz wurde kein Gerät gefunden.
4	WRITE PROTECTED *(schreibgeschützt)*	Die Schreibschutzkerbe der Diskette, auf die Sie zu schreiben versuchen, ist zugeklebt.
5	END OF DATA *(Ende der Daten)*	Sie haben versucht, nach Erreichen des Dateiendes noch einen Satz von der Datei zu lesen.
6	PATH NOT FOUND *(Pfad nicht gefunden)*	Es gibt keine Datei mit dem angegebenen Dateinamen.
7	PATH NOT FOUND *(Pfad nicht gefunden)*	Es gibt keine Datei mit dem angegebenen Dateinamen.
8	I/O ERROR *(Ein-/Ausgabefehler)*	Die Laufwerktür ist offen, oder die Diskette ist nicht für ProDOS formatiert.
9	DISK FULL *(Diskette voll)*	Auf der Diskette, auf die Sie zu schreiben versuchen, sind zu viele Dateien, oder der Speicherplatz ist aufgebraucht.
10	FILE LOCKED *(Datei schreibgeschützt)*	Sie haben versucht, auf eine schreibgeschützte Datei zu schreiben.
11	INVALID OPTION *(ungültige Option)*	Die Option, die Sie angegeben haben, gibt es bei diesem Befehl nicht.
12	NO BUFFERS AVAILABLE *(keine Puffer erhältlich)*	Der Speicher ist voll; Sie können die Datei nicht öffnen, weil kein Platz ist.
13	FILE TYPE MISMATCH *(falscher Dateityp)*	Der Befehl funktioniert bei dem von Ihnen angegebenen Dateityp nicht.
14	PROGRAM TOO LARGE *(Programm zu groß)*	Es ist nicht genügend Speicherplatz vorhanden, um das Programm zu laden.

15	NOT DIRECT COMMAND *(kein direkter Befehl)*	Der Befehl, den Sie benutzen wollen, muß in ein Programm eingebettet werden; er kann im Direktmodus nicht ausgeführt werden.
16	SYNTAX ERROR *(Syntaxfehler)*	Sie haben einen fehlerhaften Dateinamen, eine fehlerhafte Option oder ein falsches Komma bei Ihrer Anweisung eingegeben.
17	DIRECTORY FULL *(Verzeichnis voll)*	Sie können keine weitere Datei im Wurzelverzeichis anlegen. Es enthält bereits 51 Dateien.
18	FILE NOT OPEN *(Datei nicht offen)*	Sie haben versucht, auf eine geschlossene Textdatei zuzugreifen; die Datei muß zuerst geöffnet werden.
19	DUPLICATE FILENAME *(doppelter Dateiname)*	Sie haben versucht, eine Datei anzulegen oder umzubenennen, wobei der von Ihnen angegebene Pfadname schon existiert.
20	FILE BUSY *(Datei beschäftigt)*	Sie haben versucht, eine Datei zu öffnen, die schon offen ist.
21	FILE(S) STILL OPEN *(Datei(en) noch offen)*	Ihr letztes Programm hat nicht alle Dateien geschlossen, bevor es endete.

Anhang D
Applesoft-Fehlercodes

In diesem Anhang sind die Applesoft-Fehlermeldungen mit der Codenummer aufgelistet, die der Apple dabei zurückgibt. Die Fehlercodes werden an die gleiche Speicherstelle (222) zurückgegeben wie die ProDOS-Fehlercodes, und Sie müssen in dieser Liste wie auch in der Liste in Anhang C nachschlagen, um den hier gefundenen Code zu interpretieren.

Die Fehlermeldungen in dieser Liste erscheinen auf dem Bildschirm, wenn Sie in Ihrem Programm nicht einen der Applesoft- Befehle ONERR GOTO oder ONERR GOSUB benutzen. In diesem Fall tritt Ihre Fehlerbehandlungsroutine in Aktion, bevor die Meldung auf dem Bildschirm ausgegeben werden kann. Sie können sich aber den Fehlercode mit einer Anweisung wie

```
A = PEEK(222)
```

in die Variable A holen. Applesoft speichert die Nummer der Zeile, in der ein Fehler aufgetreten ist, immer in den Stellen 218 und 219. Sie können die Zeilennummer mit der Anweisung

```
FZ = PEEK(218)+(PEEK(219) * 256)
```

berechnen und in der Variablen FZ ablegen.

Es gibt zwei Fehlermeldungen, bei denen kein Fehlercode an die Stelle 222 zurückgegeben wird. Das sind die Meldungen CAN'T CONTINUE (kann nicht weitermachen) und ILLEGAL DIRECT (Direktmodus nicht erlaubt). Die Fehlermeldung CANT CONTINUE tritt auf, wenn Sie den CONT-Befehl bei einem Programm verwenden, das nach einer Unterbrechung mit Control-Reset oder der STOP-Anweisung geändert worden ist, oder bei einem Programm, das mit einem Fehler endete. Die Fehlermeldung ILLEGAL DIRECT tritt auf, wenn Sie einen Applesoft-Befehl im Direktmodus benutzen wollen, der dort nicht gültig ist (wie z. B. INPUT).

Code	Fehlermeldung
0	NEXT WITHOUT FOR (NEXT ohne FOR)
16	SYNTAX ERROR (Syntaxfehler)
22	RETURN WITHOUT GOSUB (RETURN ohne GOSUB)
42	OUT OF DATA (Daten sind ausgegangen)
53	ILLEGAL QUANTITY (illegale Größe)
69	OVERFLOW (Überlauf)
77	OUT OF MEMORY (kein Speicherplatz mehr)
90	UNDEF'D STATEMENT ERROR (nicht definierte Anweisung)
107	BAD SUBSCRIPT (falscher Index)
120	REDIM'D ARRAY (Bereich mehrmals dimensioniert)
133	DIVISION BY ZERO (Division durch Null)
163	TYPE MISMATCH (Typ stimmt nicht überein)
176	STRING TOO LONG (Zeichenkette ist zu lang)
191	FORMULA TOO COMPLEX (Formel zu komplex)
224	UNDEFINED FUNCTION (undefinierte Funktion)
254	REENTER? (falsche Antwort auf eine INPUT-Anweisung)
255	CONTROL INTERRUPT ATTEMPTED (Kontrollunterbrechung versucht)

Anhang E

ASCII-Zeichencodes

In diesem Anhang wird die ASCII-Zeichenmenge aufgelistet, die von einem Apple II benutzt wird. ASCII ist eine Abkürzung für American Standard Code for Information Interchange (Amerikanischer Standardcode zum Austausch von Informationen), dem am weitesten verbreiteten Code zum Verschlüsseln von Daten für die Kommunikation mit Computern. In der Tabelle sind alle Codes bis zur Zahl 127 aufgelistet. In Wirklichkeit gibt es 256 ASCII-Zeichen, aber Applesoft verwendet die Zeichen oberhalb von 127 zur Darstellung der reservierten Wörter bei Applesoft-BASIC (siehe Anhang F). Das wird gemacht, um Programme zu komprimieren und um Platz beim Speichern dieser Wörter zu sparen.

In der ersten Spalte dieser Tabelle steht der ASCII-Code eines Zeichens in dezimaler Schreibweise. In der zweiten Spalte steht der gleiche Code in hexadezimaler Schreibweise. Die dritte Spalte enthält das Zeichen auf der Tastatur des Apple II, das durch diesen Code repräsentiert wird. Das trifft nur für die ASCII-Zeichen ab 32 zu; bei den Zeichen unterhalb von 32 steht hier eine Abkürzung aus zwei oder drei Buchstaben, die dazu verwendet wird, auf dieses Zeichen zu verweisen. Das ASCII-Zeichen mit der Nummer 0 wird zum Beispiel als NUL, und das ASCII-Zeichen mit der Nummer 7 als BEL bezeichnet. (Bei der Anweisung PRINT CHR$(7) gibt der Lautsprecher des Apple einen Ton von sich.)

In der vierten Zeile steht das Symbol, das auf dem Bildschirm erscheint, wenn Sie sich das entsprechende ASCII-Zeichen mit einer PRINT-Anweisung ausgeben lassen. Die Zeichen unterhalb von 32 werden als Control-Zeichen bezeichnet und werden erzeugt, indem Sie die Control-Taste gleichzeitig mit einer anderen Taste drücken. Dabei erscheint normalerweise nichts auf dem Bildschirm. Das ASCII-Zeichen mit der Nummer 1 wird zum Beispiel mit den Tasten Control und A erzeugt und das Zeichen mit der Nummer 4 mit Control-D.

Obwohl es nicht nötig ist, die Zeichen der ASCII-Tabelle beim Programmieren im Kopf zu haben, kennen sich doch Leute, die viel programmie-

ren, ziemlich schnell darin aus, wo bestimmte Zeichen in der Tabelle zu finden sind. Sehr wichtig ist es aber zu wissen, daß dieses Zeichen existiert und wo man es nachschlagen kann.

dez.	**hex.**	**Zeichen**	**Control-Code**	**dez.**	**hex.**	**Zeichen**	**Bildschirm**
0	00	NUL	Control-@	32	20	space	
1	01	SOH	Control-A	33	21	!	!
2	02	STX	Control-B	34	22	”	”
3	03	ETX	Control-C	35	23	#	#
4	04	EOT	Control-D	36	24	$	$
5	05	ENQ	Control-E	37	25	%	%
6	06	ACK	Control-F	38	26	&	&
7	07	BEL	Control-G	39	27	’	’
8	08	BS	Control-H (←)	40	28	(	(
9	09	HT	Control-I (TAB)	41	29	)	)
10	0A	LF	Control-J (↓)	42	2A	*	*
11	0B	VT	Control-K (↑)	43	2B	+	+
12	0C	NP	Control-L	44	2C	,	,
13	0D	CR	Control-M (RETURN)	45	2D	-	-
14	0E	SO	Control-N	46	2E	.	.
15	0F	SI	Control-O	47	2F	/	/
16	10	DLE	Control-P	48	30	0	0
17	11	DC1	Control-Q	49	31	1	1
18	12	DC2	Control-R	50	32	2	2
19	13	DC3	Control-S	51	33	3	3
20	14	DC4	Control-T	52	34	4	4
21	15	NAK	Control-U	53	35	5	5
22	16	SYN	Control-V	54	36	6	6
23	17	ETB	Control-W	55	37	7	7
24	18	CAN	Control-X	56	38	8	8
25	19	EM	Control-Y	57	39	9	9
26	1A	SUB	Control-Z	58	3A	:	:
27	1B	ESC	Control-[	59	3B	;	;
28	1C	FS	Control-\	60	3C	<	<
29	1D	GS	Control-]	61	3D	=	=
30	1E	RS	Control-^	62	3E	>	>
31	1F	US	Control-–	63	3F	?	?

dez.	hex.	Zeichen	Bildschirm	dez.	hex.	Zeichen	Bildschirm
64	40	@	@	96	60	’	’
65	41	A	A	97	61	a	a
66	42	B	B	98	62	b	b
67	43	C	C	99	63	c	c
68	44	D	D	100	64	d	d
69	45	E	E	101	65	e	e
70	46	F	F	102	66	f	f
71	47	G	G	103	67	g	g
72	48	H	H	104	68	h	h
73	49	I	I	105	69	i	i
74	4A	J	J	106	6A	j	j
75	4B	K	K	107	6B	k	k
76	4C	L	L	108	6C	l	l
77	4D	M	M	109	6D	m	m
78	4E	N	N	110	6E	n	n
79	4F	O	O	111	6F	o	o
80	50	P	P	112	70	p	p
81	51	Q	Q	113	71	q	q
82	52	R	R	114	72	r	r
83	53	S	S	115	73	s	s
84	54	T	T	116	74	t	t
85	55	U	U	117	75	u	u
86	56	V	V	118	76	v	v
87	57	W	W	119	77	w	w
88	58	X	X	120	78	x	x
89	59	Y	Y	121	79	y	y
90	5A	Z	Z	122	7A	z	z
91	5B	[	[	123	7B	{	{
92	5C	\	\	124	7C	|	|
93	5D	]	]	125	7D	}	}
94	5E	^	^	126	7E	~	~
95	5F	_	_	127	7F	DEL	

Anhang F
Zeichen für unter Applesoft reservierte Wörter

Werden BASIC-Programme mit dem SAVE-Befehl auf eine Diskette übertragen, so werden Wörter, die unter Applesoft reserviert werden, als Zeichen abgespeichert und brauchen nur den Platz von einem Byte. Das reservierte Wort GOTO wird zum Beispiel durch das ASCII-Zeichen mit der Nummer 171 dargestellt und das Wort RETURN durch 177. Das wird gemacht, um Platz zu sparen. Wenn Sie eine BAS-Datei überprüfen (indem Sie sie mit BLOAD in den Speicher laden und sich dann mit dem Monitor oder dem PEEK-Befehl diesen Speicherbereich anschauen), werden Sie diese Codes sehen und nicht die Codes der ausgeschriebenen Befehle. Alle reservierten Wörter werden durch ASCII-Zeichen oberhalb von 127 dargestellt. Diese Codezahlen mit den von ihnen dargestellten Wörtern sind hier aufgelistet:

Zeichen	**reserviertes Wort**	**Zeichen**	**reserviertes Wort**	**Zeichen**	**reserviertes Wort**
128	END	164	LOMEM:	200	+
129	FOR	165	ONERR	201	−
130	NEXT	166	RESUME	202	*
131	DATA	167	RECALL	203	/
132	INPUT	168	STORE	204	^
133	DEL	169	SPEED=	205	AND
134	DIM	170	LET	206	OR
135	READ	171	GOTO	207	>
136	GR	172	RUN	208	=
137	TEXT	173	IF	209	<
138	PR#	174	RESTORE	210	SGN
139	IN#	175	&	211	INT

140	CALL	176	GOSUB	212	ABS
141	PLOT	177	RETURN	213	USR
142	HLIN	178	REM	214	FRE
143	VLIN	179	STOP	215	SCRN(
144	HGR2	180	ON	216	PDL
145	HGR	181	WAIT	217	POS
146	HCOLOR=	182	LOAD	218	SQR
147	HPLOT	183	SAVE	219	RND
148	DRAW	184	DEF	220	LOG
149	XDRAW	185	POKE	221	EXP
150	HTAB	186	PRINT	222	COS
151	HOME	187	CONT	223	SIN
152	ROT=	188	LIST	224	TAN
153	SCALE=	189	CLEAR	225	ATN
154	SHLOAD	190	GET	226	PEEK
155	TRACE	191	NEW	227	LEN
156	NOTRACE	192	TAB(	228	STR$
157	NORMAL	193	TO	229	VAL
158	INVERSE	194	FN	230	ASC
159	FLASH	195	SPC(	231	CHR$
160	COLOR=	196	THEN	232	LEFT$
161	POP	197	AT	233	RIGHT$
162	VTAB	198	NOT	234	MID$
163	HIMEM:	199	STEP		

Anhang G
Die MLI-Aufrufe

Dieser Anhang enthält eine Nachschlagetabelle aller MLI-Aufrufe und ihrer Parameter. Eine vollständigere Beschreibung dieser Aufrufe und ihrer Parameter ist in Kapitel 10 zu finden.

CREATE ($CO)

Parameteranzahl =	7
Pfadname	2 Bytes
Zugriff	1 Byte
Dateityp	1 Byte
Aux-Typ	2 Bytes
Speichertyp	1 Byte
Anlegedatum	2 Bytes
Anlegeuhrzeit	2 Bytes

DESTROY ($C1)

Parameteranzahl =	1
Pfadname	2 Bytes

RENAME ($C2)

Parameteranzahl =	2
Pfadname	2 Bytes
neuer Pfadname	2 Bytes

SET_FILE_INFO ($C3)

Parameteranzahl =	7
Pfadname	2 Bytes
Zugriff	1 Byte
Dateityp	1 Byte
Aux-Typ	2 Bytes
Nullfeld	3 Bytes
mod. Datum	2 Bytes
mod. Uhrzeit	2 Bytes

GET_FILE_INFO ($C4)

Parameteranzahl =	$A
Pfadname	2 Bytes
Zugriff	1 Byte
Dateityp	1 Byte
Aux-Type	2 Bytes
Speichertyp	1 Byte
belegte Blöcke	2 Bytes
mod. Datum	2 Bytes
mod. Uhrzeit	2 Bytes
Anlegedatum	2 Bytes
Anlegeuhrzeit	2 Bytes

ON_LINE ($C5)

Parameteranzahl =	2
Einheitsnummer	1 Byte
Datenpuffer	2 Byte

SET_PREFIX ($C6)

Parameteranzahl =	1
Pfadname	2 Bytes

GET_PREFIX ($C7)

Parameteranzahl =	1
Datenpuffer	2 Bytes

OPEN ($C8)

Parameteranzahl =	3
Pfadname	2 Bytes
I/O-Puffer	2 Bytes
Referenznummer	1 Byte

NEWLINE ($C9)

Parameteranzahl =	3
Referenznummer	1 Byte
Maske	1 Byte
Zeilenwechsel-Zeichen	1 Byte

READ ($CA)

Parameteranzahl =	4
Referenznummer	1 Byte
Datenpuffer	2 Bytes
Anforderungsanzahl	2 Bytes
Übertragungsanzahl	2 Bytes

WRITE ($CB)

Parameteranzahl =	4
Referenznummer	1 Byte
Datenpuffer	2 Bytes
Anforderungsanzahl	2 Bytes
Übertragungsanzahl	2 Bytes

CLOSE ($CC)

Parameteranzahl =	1
Referenznummer	1 Byte

FLUSH ($CD)

Parameteranzahl =	1
Referenznummer	1 Byte

SET_MARK ($CE)

Parameteranzahl =	2
Referenznummer	1 Byte
Position	3 Bytes

GET_MARK ($CF)

Parameteranzahl =	2
Referenznummer	1 Byte
Position	3 Bytes

SET_EOF ($D0)

Parameteranzahl =	2
Referenznummer	1 Byte
Dateiende (EOF)	3 Bytes

GET_EOF ($D1)

Parameteranzahl =	2
Referenznummer	1 Byte
Dateiende (EOF)	3 Bytes

SET_BUF ($D2)

Parameteranzahl =	2
Referenznummer	1 Byte
I/O-Puffer	2 Bytes

GET_BUF ($D3)

Parameteranzahl =	2
Referenznummer	1 Byte
I/O-Puffer	2 Bytes

GET_TIME ($82)

Parameteranzahl =	0

ALLOC_INTERRUPT ($40)

Parameteranzahl =	2
Unterbrechungsnummer	1 Byte
Unterbrechungscode	2 Bytes

DEALLOC_INTERRUPT ($41)

Parameteranzahl =	1
Unterbrechungsnummer	1 Byte

READ_BLOCK ($80)

Parameteranzahl =	3
Einheitsnummer	1 Byte
Datenpuffer	2 Bytes
Blocknummer	2 Bytes

WRITE_BLOCK ($81)

Parameteranzahl =	3
Einheitsnummer	1 Byte
Datenpuffer	2 Bytes
Blocknummer	2 Bytes

Anhang H
MLI-Fehlercodes

Dieser Anhang enthält eine Liste der Fehlercodes, die von den MLI-Aufrufen in den Akkumulator zurückgegeben werden können. Normalerweise wird der Wert $00 zurückgegeben; er zeigt den erfolgreichen Abschluß eines Aufrufs an. Jeder andere Wert zeigt einen Fehler an.

Code	**Beschreibung**
$00	kein Fehler
$01	falsche Systemaufrufnummer
$04	falsche Parameteranzahl
$25	Unterbrechungstabelle voll
$27	Ein-/Ausgabefehler
$28	kein Gerät angeschlossen
$2B	Diskette ist schreibgeschützt
$2E	Diskette ausgewechselt
$40	ungültiger Pfadname
$42	maximale Anzahl von Dateien offen
$43	ungültige Referenznummer
$44	Verzeichnis nicht gefunden
$45	Datenträger nicht gefunden
$46	Datei nicht gefunden
$47	doppelter Dateiname
$48	Diskette voll
$49	Stammverzeichnis voll
$4A	inkompatibles Dateiformat, auch bei einem Verzeichnis
$4B	nicht unterstützter Speichertyp
$4C	Dateiende angetroffen
$4D	Position außerhalb des Bereichs
$4E	Dateizugriffsfehler oder Datei schreibgeschützt
$50	Datei ist offen

$51	Verzeichnisstruktur beschädigt
$53	ungültiger Systemaufrufparameter
$55	Tabelle des Dateikontrollblocks voll
$56	falsche Pufferadresse
$57	doppelter Diskettenname
$5A	Dateistruktur beschädigt

Anhang I

Verwendung der nullten Seite (Zero Page) unter Applesoft

Stellen	Verwendung
\$00–\$05	Sprunganweisungen beim Verbleib in Applesoft.
\$0A–\$0C	Enthält die Sprungadresse der USR-Funktion (die USR-Funktion ist in Kapitel 6 beschrieben).
\$0D–\$17	Zähler für allgemeine Aufgaben und Statusbits bei Applesoft-Programmen.
\$20–\$4F	Für den Systemmonitor des Applle II reservierte Stellen. Der Bereich von \$40 bis \$4E wird vom MLI benutzt und wiederhergestellt. Der Bereich von \$3A bis \$3F wird von den Laufwerktreiberroutinen von ProDOS benutzt und nicht wiederhergestellt.
\$50–\$61	Zeiger für allgemeine Aufgaben bei Applesoft-Programmen.
\$62–\$66	Das Ergebnis der letzten vom Programm durchgeführten Multiplikation oder Division ist hier gespeichert.
\$67–-\$68	Zeiger auf den Anfang des Programms (normalerweise auf \$0801 gesetzt).
\$69–\$6A	Zeiger auf den Beginn des Speicherplatzes für einfache Variablen (ganzzahlige und reelle Variablen sowie Zeigervariablen auf Zeichenketten). Zeigt auch auf das Programmende plus ein oder zwei Bytes, wenn dieser Wert nicht mit der LOMEM- Anweisung geändert worden ist.
\$6B–\$6C	Zeiger auf den Beginn des Speicherbereichs für Feldvariablen.
\$6D–\$6E	Zeiger auf das Ende des Speicherbereichs für numerische Variablen.

$6F–$70	Zeiger auf den Anfang des Bereichs für Zeichenketten. Von hier an bis zum Speicherende sind Zeichenketten abgespeichert.
$71–$72	Allgemeiner Zeiger, der von Applesoft intern für verschiedene Aufgaben verwendet wird.
$73–$74	Größte für Applesoft verfügbare Speicherstelle plus Eins, falls dieser Wert nicht mit der HIMEM-Anweisung geändert worden ist.
$75–$76	Nummer der zur Zeit ausgeführten Programmzeile.
$77–$78	Diese Stelle wird durch eine Control-C-, STOP- oder END-Anweisung geändert. Sie enthält die Nummer der Zeile, in der das Programm unterbrochen wurde.
$79–-$7A	Zeigt auf die Stelle im Speicher, wo sich die nächste auszuführende Anweisung befindet.
$7B–$7C	Nummer der Zeile, aus der zur Zeit Daten gelesen werden. Sie ändert sich, wenn die READ-Anweisung das Ende einer Zeile überschreitet.
$7D–$7E	Zeigt auf die absolute Speicherstelle, aus der Daten gerade gelesen werden und ändert sich bei jeder READ-Anweisung.
$7F–$80	Zeiger auf die gegenwärtige Eingabequelle; wird während einer INPUT-Anweisung auf $0201 und während einer READ-Anweisung auf die Daten, die das Programm gerade liest, gesetzt.
$81–$82	Enthält den Namen der zuletzt benutzten Variablen und ändert sich immer, wenn eine neue Variable benutzt wird.
$83–$84	Zeiger auf den Wert der zuletzt benutzten Variablen. Er ändert sich immer, wenn eine neue Variable verwendet wird.
$85–$9C	Allgemeine Verwendung.
$9D–$A3	Haupt-Gleitkommaakkumulator zur Verwendung bei Rechenoperationen mit reellen Zahlen.
$A4	Für allgemeine Zwecke bei mathematischen Routinen mit Gleitkommazahlen.
$A5–$AB	Das ist der Neben-Gleitkommaakkumulator bei Rechenoperationen mit reellen Zahlen.
$AC–$AE	Kennzeichenbits und Zeiger für allgemeine Verwendungszwecke.
$AF–$B0	Zeiger auf das Programmende (wird durch LOMEM nicht geändert).

$B1–$C8	Das ist die CHRGET-Routine; Applesoft ruft diese Routine immer dann auf, wenn es ein neues Zeichen für die Eingabe verlangt. In den Bytes $B8 und &B9 wird ein Zeiger auf das letzte von dieser Routine eingelesene Zeichen gespeichert.
$C9–$CD	Zufallszahl; dieser Wert wird hier von der Applesoft-Funktion RND abgelegt.
$D0–$D5	Zeiger für Notizen bei der hochauflösenden Grafik, der intern von den Grafikroutinen benutzt wird.
$D8–$DF	ONERR-Zeiger, die auch für Notizen oder zum Zwischenspeichern verwendet werden.
$E0–$E2	X- und Y-Koordinaten bei der hochauflösenden Grafik.
$E4	Farbbyte der hochauflösenden Grafik, das mit der HCOLOR-Anweisung gesetzt wird. Es legt fest, welche Farbe der nächste gemalte Punkt haben wird.
$E5–$E7	Für allgemeine Zwecke bei den Routinen der hochauflösenden Grafik.
$E8–$E9	Zeiger auf den Anfang einer Shape-Tabelle.
$EA	Kollisionszähler, der von den Applesoft-Routinen für die hochauflösende Grafik verwendet wird.
$F0–$F3	Kennzeichenbits für allgemeine Verwendungszwecke.
$F4–$F8	ONERR-Zeiger zur Verwendug bei den Befehlen ONERR GOTO und ONERR GOSUB.

Anhang **J**

Speicheraufteilung unter ProDOS

Dezimal	Hex	Bereich	
65535	$FFFF		
		Laufwerks-Routinen ProDOS-Datenbereich	
57344	$E000	ProDOS MLI	
		Reservierter Bereich	
53248	$D000		
		System I/O	
49152	$C000		
48896	$BF00	System-Global-Seite	
			Bereich für BASIC-Programme ($0C00–$BF00)
24576	$6000		
		Seite 2 der hochauflösenden Grafik	
16384	$4000		
		Seite 1 der hochauflösenden Grafik	
8192	$2000		
3840	$0C00		
		Seite 2 für Text und niedrigauflösende Grafik	
2048	$0800		
		Seite 1 für Text und niedrigauflösende Grafik	
1024	$0400		
		Nullte Seite, Stapelspeicher, Eingabe-Puffer	
0	$0000		

Bei einem auf 64K RAM erweiterten Apple II+ befindet sich der Bereich von $D000 bis $FFFF auf der Speichererweiterungs-Karte.

Anhang K
Die System-Global-Seite

Die System-Global-Seite besteht aus einem Speicherbereich, der von ProDOS für die gesamte Kommunikation mit dem MLI benutzt wird. Sie enthält einen Großteil der Daten, die für einen reibungslosen Ablauf Ihres Systems erforderlich sind. Wenn Sie sie verändern, kann das System zusammenbrechen, und Sie müssen es neu laden. In der folgenden Tabelle sind die Speicherstellen der System-Global-Seite mit den Informationen angegeben, die darin gespeichert werden.

Stelle	**Verwendung**
$BF00–$BF02	Das ist der Eintrittspunkt zu den MLI-Aufrufen. Er enthält eine JMP-Anweisung zur Stelle $BFB7.
$BF03–$BF05	Diese Bytes enthalten einen kleinen Vektor, der für zukünftige Verwendungszwecke von ProDOS reserviert ist.
$BF06–$BF08	Diese Bytes enthalten eine JMP-Anweisung zu der Uhr-/Kalender-Routine.
$BF09–$BF0B	Diese Bytes enthalten eine JMP-Anweisung zur Berichterstattung von Fehlern.
$BF0C–$BF0E	Diese Bytes enthalten eine JMP-Anweisung, die benutzt wird, wenn das System nicht in der Lage ist weiterzumachen (Systemfehler).
$BF0F	Hier befindet sich der Fehlercode (wenn kein Fehler aufgetreten ist, enthält dieses Byte den Wert 0).
$BF10–$BF1F	Dieser Bereich enthält eine Serie von zwei Bytes langen Zeigern, die beim Zugriff auf Laufwerk 1 an einem bestimmten Steckplatz benutzt werden. Diese Zeiger repräsentieren Steckplatz 0 bis 7 für den Fall, daß dort Laufwerke angeschlossen sind ($BF10–$BF11 repräsentieren Steckplatz 0,

	Laufwerk 1; $BF12–$BF13 Steckplatz 1, Laufwerk 1 usw.).
$BF20–$BF2F	Dieser Bereich besteht aus einer Serie von zwei Bytes langen Zeigern, die beim Zugriff auf ein Laufwerk mit der Laufwerknummer 2 benutzt werden. Die Zeiger repräsentieren dabei die Steckplätze 0 bis 7 ($BF20–$BF21 repräsentieren Steckplatz 0, Laufwerk 2; $BF22–$BF23 Steckplatz 1, Laufwerk 2 usw.)
$BF30	Hier werden Laufwerk- und Steckplatznummer des Gerätes festgehalten, auf das zuletzt zugegriffen wurde.
$BF31	Hier wird die Anzahl der Zugriffe auf ein Diskettengerät in dieser Sitzung minus Eins festgehalten.
$BF32–$BF3F	Diese Bytes enthalten die Suchliste aller aktiven Geräte des Systems.
$BF40–$BF4A	Diese Bytes werden für Apples Copyright-Meldung verwendet.
$BF50–$BF57	Die Aufgabe dieser Bytes ist zu diesem Zeitpunkt nicht bekannt.
$BF58–$BF6F	Dieser Bereich enthält die System-Bitabbildung. Diese wird dazu verwendet, die Speicherbereiche aufzuzeichnen, die sich im Gebrauch befinden.
$BF70–$BF7F	Dieser Bereich besteht aus 8 zwei Bytes langen Zeigern auf die Adressen der ProDOS-Dateipuffer.
$BF80–$BF87	Dieser Bereich besteht aus 4 zwei Bytes langen Zeigern auf die Unterbrechungsvektoren.
$BF88–$BF8F	Die Werte des A-, X-, Y-, Stack- und des Statusregisters werden hier während einer Unterbrechung festgehalten.
$BF90	Hier wird das Datum gespeichert.
$BF92	Hier wird die Uhrzeit gespeichert.
$BF94	Die von den Aufrufen OPEN, FLUSH und CLOSE verwendete Dateistufenzahl wird hier gespeichert.
$BF95	Dieses Byte signalisiert ProDOS die Benutzung des Backup-Bits.
$BF96–$BF97	Diese Bytes sind unbenutzt.
$BF98	Dieses Byte dient zur Identifizierung der Maschine.
$BF99	Dieses Byte sagt ProDOS, welche Steckplätze von Karten belegt sind, auf denen ROM installiert ist.
$BF9A	Mit diesem Byte wird bestimmt, ob ein Präfix

	verwendet wird. Wenn Null in diesem Byte steht, wird kein Präfix benutzt.
$BF9B	Dieses Byte zeigt an, ob das MLI aktiv ist. Wenn Bit 7 auf Eins gesetzt ist, ist das MLI aktiv.
$BF9C–$BF9D	Diese Bytes enthalten die Rückkehradresse des letzten MLI-Aufrufs.
$BF9E	In diesem Byte wird der Wert des X-Registers während eines MLI-Aufrufs zwischengespeichert.
$BF9F	Hier wird der Wert des Y-Registers während eines MLI-Aufrufs zwischengespeichert.
$BFA0–$BFFB	Dieser Bereich enthält Routinen zum Umschalten zu einer Speicherbank auf der Language-Karte.
$BFFC	In diesem Byte steht die früheste Versionsnummer des frühesten MLI, mit dem das jetzige Systemprogramm zusammenarbeiten kann.
$BFFD	Dieses Byte enthält die Versionsnummer des laufenden Systemprogramms.
$BFFE	Dieses Byte enthält die Nummer der kleinsten kompatiblen MLI-Version.
$BFFF	Dieses Byte enthält die neueste Versionsnummer von ProDOS.

Anhang L
Die ProDOS User's Disk auf einem Apple II-kompatiblen Computer lauffähig machen

Die Version der ProDOS User's Disk vom 1. Januar 84 läuft nur auf den Originalgeräten Apple II+, Apple IIe und Apple IIc. Sie enthält nämlich eine Routine zur Überprüfung, ob im „Nur Lese"-Speicher (ROM) des Computers die Zeichenfolge Apple][steht, was nur bei einem Gerät der Firma Apple erlaubt ist. Wenn Sie einen Apple II-kompatiblen Computer mit Applesoft-BASIC im ROM besitzen, können Sie durch eine kleine Modifikation der ProDOS User's Disk die obengenannte Prüfroutine ausschalten und damit in der Regel das Betriebssystem ProDOS auf Ihrem Computer zum Laufen bringen. Führen Sie dazu die folgenden Schritte aus (die zusätzlichen Erklärungen brauchen Sie nicht unbedingt zu verstehen):

1. Stellen Sie mit einem DOS-Kopierprogramm (z. B. COPYA) eine Kopie der ProDOS User's Disk her.
2. Legen Sie eine DOS 3.3-Diskette (z. B. die DOS 3.3 System Master) ins Laufwerk, um in den Applesoft-BASIC-Modus zu kommen.
3. Tippen Sie im BASIC-Modus CALL −151, und drücken Sie die Return-Taste. Links vor dem aufleuchtenden Cursor sehen Sie einen Stern: Sie befinden sich im Monitormodus. (Wenn Sie verstehen wollen, wie die folgenden Programmschritte funktionieren, können Sie Kapitel 11, Abschnitt 2 lesen. Dort wird der Monitormodus erklärt.)
4. Tauschen Sie die DOS 3.3-Diskette im Laufwerk gegen die Kopie der ProDOS User's Disk aus.

5. Tippen Sie die folgenden Anweisungen ein, und drücken Sie hinter jeder Zeile die Return-Taste:

```
0300:20 E3 03 20 D9 03 60
B7EB:00 01 09
B7F0:00 20
B7F4:01
0300G
```

Wenn der Lautsprecher piepst, haben Sie etwas falsch gemacht. Vielleicht haben Sie die Leerzeichen nicht richtig eingetippt oder die Zahl 0 mit dem Buchstaben O verwechselt. Dann müssen Sie Schritt 5 noch einmal wiederholen. Durch die Anweisung 0300G in der letzten Zeile wird das Maschinensprache-Programm aktiviert, das Sie beim Eintippen der ersten Zeile hinter der Stelle 0300 im Speicher des Computers abgelegt haben. (Im Monitormodus werden alle Werte in hexadezimaler Form angegeben. Eine kurze Erklärung der Hexadezimaldarstellung finden Sie im ersten Abschnitt von Kapitel 6.) Das Laufwerk setzt sich kurz in Bewegung, und Sektor 9 der Spur 1 auf der ProDOS User's Disk wird in die zwanzigste Speicherseite geladen (eine Speicherseite besteht aus 0100 Bytes). Im folgenden Schritt 6 überprüfen wir, ob diese Operation erfolgreich durchgeführt worden ist.

6. Tippen Sie 2000, und drücken Sie die Return-Taste. Jetzt muß auf dem Bildschirm die Zeile 2000–A9 erscheinen. Drücken Sie zur Sicherheit noch einmal die Return-Taste. Jetzt sehen Sie

   ```
   09 02 85 0C D0 11 A0
   ```

   Wenn das nicht stimmt, haben Sie wahrscheinlich die falsche Diskette eingelegt, und Sie müssen noch einmal von vorne anfangen.

7. Tippen Sie die folgenden drei Anweisungen ein, und drücken Sie hinter jeder Zeile die Return-Taste.

```
205B:38 EA
B7F4:02
0300G
```

Sobald Sie damit fertig sind, setzt sich das Laufwerk wieder kurz in Bewegung. Sie hören einen Piepston, der Sie diesmal nicht zu stören braucht.

8. Tippen Sie PR#6, und drücken Sie die Return-Taste.

Wenn alles richtig verlaufen ist, erscheint nach kurzer Aktivität des Laufwerks das Hauptmenü der System-Dienstprogramme der ProDOS User's

Disk auf dem Bildschirm. Sie können diese ProDOS-Diskette jetzt benutzen und die Wahlmöglichkeiten im Hauptmenü ausprobieren.

Es folgt noch eine kurze Beschreibung der drei Anweisungen in Schritt 7: Durch die Anweisung in der ersten Zeile werden bei dem Sektor, den Sie in den Speicher geladen haben, die Werte an den Stellen 205B und 205C in 38 bzw. EA geändert. Damit wird die Prüfroutine für den Firmennamen APPLE][lahmgelegt. In der zweiten Zeile wird der Wert an der Stelle B7F4, den Sie vorher auf 1 gesetzt haben, auf 2 abgeändert. Wenn Sie dann Ihre Maschinensprache-Routine mit der Anweisung in der letzten Programmzeile starten, wird der Sektor nicht gelesen, sondern auf die Diskette zurückgeschrieben.

Stichwortverzeichnis

SYBEX

Das ProDOS Handbuch

Arbeiten Sie mit einem Apple oder Apple-Kompatiblen? Dieses Buch wurde speziell auf Ihre Bedürfnisse zugeschnitten: Es zeigt Ihnen alles Wissenswerte über ProDOS, ein leistungsfähiges Betriebssystem für Ihren Rechner.

Die Autoren machen Sie Schritt für Schritt mit ProDOS vertraut. Zunächst erhalten Sie einen leichten Einstieg in das System durch den Vergleich zwischen DOS 3.3 und ProDOS. Dann werden Ihnen ProDOS-Routinen auf dem Apple IIe und IIc vorgestellt und erläutert – bis hin zur detaillierten Beschreibung von Maßnahmen, wie Sie wirklich alles aus diesem starken Betriebssystem herausholen.

Dabei ist es egal, auf welcher Wissensstufe Sie sich befinden. **Anfänger,** die ProDOS nutzen wollen, ohne selbst zu programmieren, lernen

- Disketten formatieren,
- Dateien erstellen, kopieren und löschen,
- Dateisysteme aufbauen.

Fortgeschrittene Anwender gebrauchen das Buch, um

- mit ProDOS in BASIC zu programmieren,
- ProDOS für Anwendungen mit Grafik und Sound einzusetzen,
- Maschinen- oder Assemblerprogramme schneller, flexibler und leistungsfähiger zu machen.

Zahlreiche informative Anhänge ergänzen dieses umfassende Handbuch, das Ihnen hilft, Ihr Apple-System mit ProDOS noch effizienter zu machen.

Über die Autoren

Karen und Timothy Rice haben jahrelange Erfahrungen in der Arbeit mit Mini- und Mikrocomputern und betreiben zusammen eine Firma für Software-Entwicklung.

ISBN 3-88745-617-3